POLITECNICO DI TORINO

III Facoltà di Ingegneria

Corso di Laurea in Ingegneria Elettronica

Tesi di Laurea

Progetto e realizzazione di un ambiente didattico di laboratorio per esercitazioni sui "sistemi elettronici digitali"

Relatore:
prof. Luciano Lavagno

Candidato:
Luca Necchi

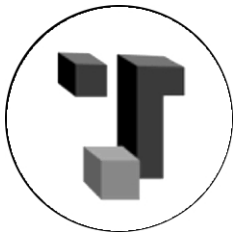
Settembre 2003

UNIVERSIDAD POLITÉCNICA DE MADRID
ESCUELA TÉCNICA SUPERIOR DE INGENIEROS DE TELECOMUNICACIÓN

Facoltà di Ingegneria

Tesi di Laurea

Progetto e realizzazione di un ambiente didattico di laboratorio per esercitazioni sui "sistemi elettronici digitali"



Relatori:
Juan Manuel Montero Martìnez
D. Octavio Nieto-Taladrìz Garcìa

Candidato:
Luca Necchi

Settembre 2003

Sommario

Scopo

Scopo di questo documento è illustrare le varie fasi di progettazione dell'ambiente didattico relativo al "*Laboratorio de Sistemas Electrònicos Digitales*", corso che viene proposto agli studenti dell' *Escuela Técnica Superior de Ingenieros de Telecomunicación* dell'*Universidad Politécnica de Madrid*.

Ambiente didattico

Lo svolgimento di detto laboratorio prevede il progetto e la realizzazione di un sistema di elaborazione numerica dei segnali. Compito dello studente sarà:

- programare un microprocessore, affinchè esso realizzi il sistema richiesto di elaborazione numerica dei segnali.
- realizzare un'apposita circuiteria di conversione analogico/digitale e digitale/analogico.

Negli anni scorsi, questo laboratorio è stato effettuato con l'ausilio di una scheda didattica basata sul processore *Motorola MC68000*.

Problema

La rapida evoluzione dei sistemi basati su microprocessori digitali e le sempre maggiori prestazioni richieste a questo tipo di dispositivi hanno dettato la necessità di aggiornare l'*hardware* di cui erano dotate le postazioni di laboratorio e, in particolar modo, il sistema di elaborazione dati utilizzato.

Soluzione proposta

È stato pertanto deciso di utilizzare, come componente centrale per lo svolgimento di questo laboratorio, la scheda denominata *Antares* recentemente creata dai ricercatori e dai tesisti della sezione "*LSI*" (laboratorio di Sistemi Integrati) del dipartimento di Ingegneria Elettronica dell'Università di Madrid.

La scheda *Antares* è basata sul microcontrollore *Motorola Coldfire MCF5272*, caratterizzato da eccellenti prestazioni e orientato ai sistemi "embedded".

Lavoro svolto

Nel corso di questa tesi e' stato realizzato un sistema completo di sviluppo software per la nuova piattaforma *Antares*.

In particolare, mi soffermerò a descrivere:

- Il programma **Ed68k**. Questo programma è stato pensato per offrire all'utente un'interfaccia di lavoro grafica ed amichevole, con la quale poter sviluppare software e poter interagire agilmente con la scheda *Antares*.
- Il **compilatore** e l'**assemblatore**. È stato necessario dotare il sistema di sviluppo software di strumenti di compilazione capaci di produrre file binari specifici per l'architettura del processore *Motorola Coldfire MCF5272*.
- Il *firmware* della scheda *Antares* denominato "**dBUG**". È stato necessario adattare questo software, creato dai ricercatori *Motorola*, alle specifiche *hardware* della scheda *Antares*.

- Il sistema operativo μ**Clinux**. Verrà descritta l'installazione e la configurazione del μ*Clinux* che é una distribuzione del sistema operativo *Linux* orientata ai sistemi "embedded".

Conclusioni

Le varie componenti software e hardware, descritte in questa tesi, sono state testate verificandone l'effettivo funzionamento e la corrispondenza con le specifiche di progetto.

Per poter avviare le esercitazioni di laboratorio su questo nuovo ambiente didattico si dovrà attendere la realizzazione, da parte dell'"*LSI*" (Laboratorio di Sistemi Integrati), di un numero adeguato di schede *Antares*, *hardware* che si sta dimostrando, per le sue prestazioni e la sua versatilità, un componente essenziale per la didattica e la ricerca presso il dipartimento di Ingegneria Elettronica dell'Università di *Madrid*.

Una linea di sviluppo futura potrebbe richiedere l'aggiunta del supporto multilingue, al fine di poterlo proporre anche agli studenti del *Politecnco di Torino*.

A mio nonno
Antonio Rigorini

A mio nonno
Antonio Rigorini

Ringraziamenti

Desidero ringraziare i miei genitori che mi hanno donato le possibilità per raggiungere questo importante traguardo.

Ringrazio tutte le persone della mia famiglia, in particolare i nonni, gli zii ed i miei fratelli.

Ringrazio tutti i professori e le persone con cui ho collaborato presso l'Università di Madrid nel corso della realizzazione di questa tesi, per essere stati dei cari amici, oltre che degli ottimi insegnanti e colleghi di lavoro.

Ringrazio Marco e tutti gli amici e le amiche che mi sono sempre stati vicini.

Ringrazio Elena.

Indice

1 Descrizione del progetto **1**

1.1 Il laboratorio LSED 1

1.2 Funzionamento del programma Ed68k 2

1.3 La scelta del compilatore 4

1.4 Il compilatore della GNU. 4

1.5 L'ambiente CYGWIN 5

1.6 Il dBUG 6

1.7 Sistema operativo μClinux 6

1.8 Struttura della distribuzione μClinux 8

2 Il programma Ed68k **10**

2.1 Il laboratorio LSED 10

2.2 Funzionamento del programma Ed68k 11

2.3 Descrizione delle funzioni del programma Ed68k 13

2.3.1 File 13

2.3.2 Modifica 14

2.3.3 Connessione 15

2.3.4 Compilazione 16

2.3.5 Esecuzione 19

2.3.6 Registri: 20

2.3.7 Memoria: . . . 22
2.3.8 Breakpoint . . . 26
2.3.9 Visualizza . . . 28
2.3.10 Finestra . . . 30
2.3.11 Help . . . 32
2.3.12 Watch: . . . 33
2.4 Programmare in C . . . 34

3 Struttura del programma Ed68k **35**
3.1 La scelta del compilatore . . . 35
3.2 Il compilatore della GNU. . . . 36
3.3 L'ambiente CYGWIN . . . 36
3.4 Le modifiche al compilatore "gcc-m68k" . . . 37
3.5 Compilazione . . . 38
3.5.1 Installazione del compilatore gcc-m68k . . . 38
3.5.2 Aggiornamento delle librerie di Cigwin . . . 39

4 Il firmware dBUG **40**
4.1 Cosa è il dBUG . . . 40
4.2 Struttura del dBUG. . . . 41
4.3 Modifiche al dBUG . . . 41
4.4 Utilizzo del dBUG . . . 42

5 Installazione del sistema operativo μClinux **47**
5.1 Sistema operativo . . . 47
5.1.1 μClinux . . . 47
5.2 Architettura software del μClinux . . . 49
5.2.1 Schema di installazione. . . . 49
5.3 Utility di sviluppo . . . 51
5.3.1 Installazione . . . 52
5.4 Il boot loader CoLiLo . . . 53

5.4.1 Compilazione 54
5.4.2 Istruzioni d'uso 56
5.4.3 Descrizione tecnica 60
5.5 Il sistema operativo μClinux 60
5.5.1 Struttura della distribuzione 61
5.5.2 Modifiche alla distribuzione 62
5.5.3 Compilazione della distribuzione 63
5.6 Filesystem di μClinux 67
5.6.1 "filesystem" RAMFS 67
5.6.2 "filesystem" NFS 67
5.6.3 "filesystem" ROMFS 68
5.6.4 "filesystem" JFFS 68
5.7 Lavorando con μClinux 68
5.7.1 Configurazione dell'interfaccia Ethernet 68
5.7.2 Accesso a directory remote tramite NFS 69
5.7.3 Uso di un filesystem JFFS 70
5.7.4 Aggiornamento del μClinux 70

6 Conclusioni **72**

A Descrizione della piattaforma Antares **73**
A.1 Microprocessore 75
A.1.1 Caratteristiche del *MCF5272* 77
A.2 Memoria 79
A.2.1 Memoria SDRAM 80
A.2.2 Memoria Flash 80
A.3 Interfacce 81
A.3.1 Interfaccia Ethernet 82
A.3.2 Interfaccia USB 84
A.3.3 Interfaccia seriale RS-232 84

A.3.4 Entrate/uscite digitali 85
A.3.5 Interfaccia *BDM*/*JTAG* 87
A.3.6 Bus di espansione 87
A.4 Alimentazione 89

B Bibliografia **91**

C Schemi elettrici di *Antares* **93**

Elenco delle figure

1.1 Schermata iniziale del programma Ed68k 3

2.1 Schermata iniziale del programma Ed68k 12
2.2 Menu Archivo 13
2.3 Menu Editar 14
2.4 Menù Conexiòn 15
2.5 Configurazione porta seriale 15
2.6 Menù Construir 16
2.7 Configurazione parametri del linker 16
2.8 Programma caricato sulla scheda 18
2.9 Menù Ejecutar 19
2.10 Evento Reset 19
2.11 Menù Registros 20
2.12 Modifica Registri 20
2.13 Menù Memoria 22
2.14 Visualizzare posizione di memoria 22
2.15 Modificare posizione di memoria 23
2.16 Cerca dato 23
2.17 Copiare blocco di memoria 24
2.18 Riempire blocco di memoria 24
2.19 Visualizza blocco di memoria 25

2.20 Visualizza mappa di memoria 25
2.21 Menù P.Ruptura 26
2.22 Aggiunta di un breakpoint 26
2.23 Visualizzare la lista di breakpoint 27
2.24 Menù Ver 28
2.25 Barra di stato 28
2.26 Barra di tools 28
2.27 Barra dei registri 29
2.28 Convertitore Binario, Decimale, Esadecimale 29
2.29 Menù Ventana 30
2.30 Terminale 31
2.31 Menù Ayuda 32
2.32 Finestra “Acerca” 32
2.33 Menù Watch 33
2.34 Finestra di Watch 33
2.35 Finestra di Watch 34

4.1 Prompt del firmware dBUG 41
4.2 Mappa di memoria 44
4.3 Parametri di configurazione 45

5.1 Mappa di memoria Antares 60
5.2 Schermata di configurazione del μClinux 63
5.3 Schermata di configurazione del *uclinux*. 65
5.4 Schermata di configurazione del kernel di μClinux 66

A.1 Piattaforma di sviluppo Antares 74
A.2 Modelli della famiglia Coldfire della Motorola 76
A.3 Diagramma a blocchi del MCF5272 77
A.4 Dettaglio della memoria SDRAM 80
A.5 Dettaglio della memoria Flash 81

A.6 Dettaglio dell'interfaccia Ethernet 82
A.7 Diagramma a blocchi dell'interfaccia Ethernet 83
A.8 Detaglio dell'interfaccia USB 84
A.9 Segnali dell'interfaccia RS232 85
A.10 Dettaglio delle interfacce RS-232 85
A.11 Dettaglio delle porte di entrata/uscita 86
A.12 Dettaglio dell'interfaccia BDM/JTAG 87
A.13 Segnali del bus di espansione 88
A.14 Dettaglio del bus di espansione 89
A.15 Dettaglio dello stadio di alimentazione 90

Capitolo 1

Descrizione del progetto

1.1 Il laboratorio LSED

Nel corso denominato *LSED (Laboratorio di Sistemi Elettronici Digitali)*, gli studenti devono progettare e costruire alcuni circuiti da connettere alle porte di ingresso/uscita della scheda *Antares* e programmare la scheda stessa, in linguaggio *Assembler* o *C*, al fine di ottenere il funzionamento richiesto dal tema dell'esercitazione. Solitamente si tratta di creare un sistema di elaborazione numerica dei segnali, che permetta di:

- amplificare e attenuare
- ritardare
- filtrare, potendo variare i coefficienti di una funzione di trasferimento assegnata.

Per fare ciò, sarà necessario connettere alle porte di ingresso/uscita dei convertitori analogico/digitale e digitale/analogico, che saranno pilotati dalla stessa scheda *Antares* tramite appositi segnali.

L'esercitazione potrà poi essere arricchita aggiungendo componenti e funzionalità al sistema. Ecco alcune delle estensioni possibli.

- Visualizzare dati, relativi al processamento in esecuzione, su di un display *LCD* a 7 segmenti.

- Effettuare il *controllo remoto* del programma, con un comando ad infrarossi o a radiofrequenza.
- Effettuare il *controllo remoto* del programma, con l'ausilio di una calcolatrice HP48, HP49, di un calcolatore palmare o di un portatile.
- Aggiungere funzionalità di *riconoscimento vocale*.
- Aggiungere funzionalità di *sintesi vocale*.
- Implementare un semplice *frequenzimetro digitale*.
- Implementare un semplice generatore di funzioni programmabile.

1.2 Funzionamento del programma Ed68k

Il programma *Ed68k* svolge pertanto il ruolo di interfaccia utente, permettendo di effettuare tutte le operazioni richieste senza dover conoscere esattamente l'hardware e le funzionalità della scheda *Antares* e senza doversi preoccupare della complicata sintassi necessaria per pilotare il compilatore e il firmware "dBUG".

Ecco i compiti principali del programma *Ed68k*.

- Fornire un *editor* dove sia possibile scrivere codice in C o in Assembler.
- *Compilare* il codice editato e segnalare gli errori. Questa parte sarà approfondita nella sezione 2.3.4.
- *Inviare* alla scheda i file binari compilati.
- Permettere il *debug* del software, tramite i comuni tool di esecuzione *passo a passo*, *watch* di variabili e registri, *breakpoint* .... Questa parte sarà approfondita nella sezione 2.3.5 e seguenti.
- Presentare un *terminale* per ricevere e inviare dati al software in esecuzione sulla scheda, tramite la connessione seriale.
- Gestire le principali situazioni di *errore* che si possono verificare o intercettare la pressione del tasto *RESET* posto sulla scheda.

- Modificare in maniera agile i *registri* del processore.
- Modificare la *memoria* della scheda e il contenuto delle *variabili*.

Appena avviato, il programma *Ed68k* si presenta come visualizzato nella ***Figura 1.1***

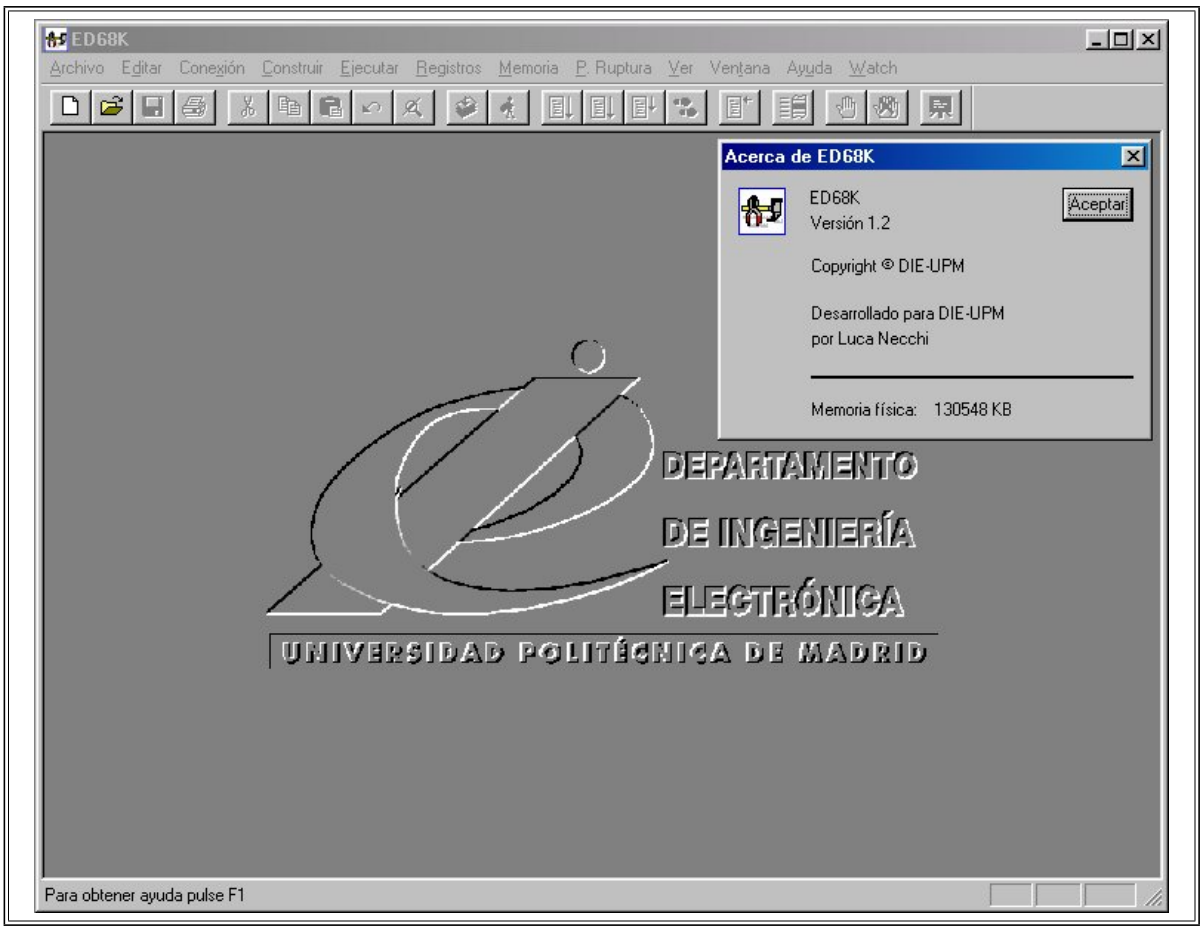
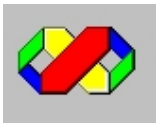


Figura 1.1: *Schermata iniziale del programma Ed68k*



Il programma *Ed68k* è stato scritto principalmente in *C++* con Microsoft *Visual Studio 6* e fa largo uso delle classi e librerie grafiche proprietarie messe a disposizione da questo tool. Il programma è completato da alcuni file batch e da una versione del noto compilatore della *GNU* modificata per produrre file binari adatti a lavorare sulla scheda *Antares*.

1.3 La scelta del compilatore

Per poter eseguire programmi sulla piattaforma *Antares* occorre che questi siano stati compilati usando un compilatore specifico. In particolare il compilatore deve presentare le seguenti caratteristiche:

- deve produrre file binari per un processore *MCF5272* della famiglia *Motorola Coldfire*, processore che è stato ampiamente descritto nella sezione A.1.
- deve funzionare in un ambiente Windows, in quanto il programma *ed68k* lavora in tale ambiente.
- deve essere utilizzabile tramite "linea di comando".
- deve essere gratuito o molto economico, in quanto una copia di tale compilatore deve essere installata su ogni computer del laboratorio dove verranno impartite le esercitazioni del corso *LSED*[1].

Analizzati tali requisiti, è stato scelto di utilizzare il noto compilatore della *GNU*. Nel seguito viene descritto questo compilatore e viene spiegato come poterlo installare in un ambiente *Windows*. Nel seguito, indicheremo questo compilatore con il nome "gcc-m68k" per non confonderlo con il compilatore di sistema.

1.4 Il compilatore della GNU.

In una qualsiasi distribuzione di *Linux* è possibile trovare il compilatore *GCC*, ovvero il Gnu C Compiler, l'assemblatore *AS*, il linker *LD*. Questi programmi fanno parte dei due pacchetti *gcc* e *binutils* entrambi distribuiti sotto licenza *GPL*, il che vuol dire che il codice ivi contenuto può essere liberamente distribuito, utilizzato e modificato, a patto di ridistribuire nuovamente il codice sotto licenza *GPL*.

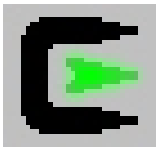
Una licenza di questo tipo rende possibile eseguire detto codice su qualsiasi architettura di sistema e qualsiasi sistema operativo, in quanto la presenza del codice sorgente permette di adattare totalmente il programma alle nostre esigenze. Le

[1] "Laboratorio de Sistemas Electrònicos Digitales"

operazioni da svolgere nel nostro caso specifico, per adattare il compilatore, sono le seguenti:

- modificare, tramite l'uso di alcune patch, il codice sorgente del compilatore, affinchè supporti l'architettura *ColdFire*.
- compilare tale codice per un ambiente Windows.

1.5 L'ambiente CYGWIN



Cygwin è un ambiente di lavoro *Unix*, che si installa su PC con sistema operativo Windows. È possibile reperire questo software all'indirizzo internet [11]. La procedura di installazione di *Cygwin* è molto semplice. Sarà sufficiente scaricare dalla rete il programma di *setup* e, grazie ad un interfaccia grafica, selezionare i pacchetti desiderati. Il setup stesso provvederà in seguito a scaricare, dal mirror selezionato, i soli pacchetti scelti. Per le nostre esigenze sarà necessario aggiungere alla configurazione di default i pacchetti "devel" dove vi sono il compilatore, l'assemblatore e il programma " patch".

Una volta installato *Cygwin* sul proprio *PC*, potremo utilizzare una console "bash" ben nota a qualsiasi utente *Linux* e da questa effettuare tutte le operazioni tipiche di una console *Linux*, nonostante il sistema operativo del PC sia Windows.

Il pacchetto *Cygwin* permette inoltre di installare molteplici tool grafici, come il server *XFree*, e di conseguenza utilizzare applicativi grafici tipici di un sistema *Linux*. Per i nostri scopi non sarà necessaria nessuna di queste funzionalità.

È importante notare che il sistema *Cygwin* funziona grazie ad alcune librerie che sono memorizzate nella sottodirectory */BIN* nel path dove abbiamo installato *Cygwin* e che vengono aggiunte alla variabile "PATH" di sistema solamente durante l'esecuzione di *Cygwin*. Aggiungendo questa directory al path di sistema saremo in grado di utiizzare gli applicativi *unix* offerti da *Cygwin* anche senza lanciarli dalla console "bash". Questo risulterà molto importante per noi, in quanto sarà lo stesso programma *Ed68k* a dover effettuare chiamate al compilatore.

1.6 Il dBUG

Il dBUG è un firmware prodotto da *Motorola* e destinato a molte delle schede dimostrative su cui sono installati i processori *Motorola*

Questo firmware, descritto nel capitolo 4, offre un'interfaccia testuale che può essere usata per scaricare, eseguire codice e gestire le funzioni basilari di un ambiente di "debug".

L'interfaccia utente è di tipo "linea di comando" e per il suo utilizzo si dovrà utilizzare un terminale seriale.

L'attuale versione del dBUG supporta le seguenti famiglie di microcontrollori:

- M68000
- ColdFire MCF5200
- PowerPC

Il codice sorgente di questo firmware viene distribuito liberamente sul sito internet [13].

1.7 Sistema operativo μClinux

Il sistema operativo *μClinux* è una versione del popolare sistema operativo *Linux* orientata ai microcontrollori che non dispongano di un'unità di gestione della memoria (*MMU*), come nel caso del *MCF5272*. Questo sistema operativo è caratterizzato dall'occupare uno spazio decisamente più modesto di quello di una convenzionale distribuzione *Linux*, fatto che lo rende idoneo allo sviluppo di sistemi embedded. A titolo di esempio, lo spazio occupato da un *kernel μClinux* configurato con le opzioni standard, può essere minore di *500 KBytes*, mentre quello di una distribuzione completa *μClinux*, fornita dei più comuni applicativi, può essere inferiore ai *900 KBytes*.

Come nel caso di molte altre distribuzioni di *Linux*, *μClinux* viene distribuito sotto licenza *GPL*. Questo significa che l'intero sistema operativo è accompagnato dal relativo codice sorgente, modificabile e ridistribuibile nei termini della licenza *GPL* consultabile in [18]. Ciò è chiaramente uno dei maggiori vantaggi offerti da

questo *sistema operativo*, in quanto ci permette di personalizzare completamente il sistema in modo che si adatti esattamente alle nostre necessità.

A quanto detto, bisogna aggiungere che il *kernel* di *Linux* offre al nostro sistema i notevoli vantaggi tipici di un qualsiasi computer dotato di *Linux*, ovvero:

- si tratta di un sistema operativo multitasking.
- è dotato di un'architettura modulare, che permette di configurare il kernel con le sole opzioni necessarie, riducendo lo spazio occupato.
- offre il supporto di molteplici protocolli di rete: *TCP/IP*, *PPP*, *SLIP*, etc.
- offre il supporto di molteplici fyle system: *NFS*, *Ext2*, *FAT32*, *FAT16*, etc.
- é robusto e affidabile.
- la disponibilità del codice sorgente permette di adattarsi alle necessità concrete di qualsiasi applicazione.
- é gratuito, dovuto al fatto che gran parte del software è distribuito sotto licenza *GPL*.
- permette la creazione di driver per supportare nuovo *hardware*. Questo punto è facilitato ulteriormente dall"esistenza di un gran numero di driver "open source", che possono essere pertanto adattati.

La *API*[2] del sistema operativo *μClinux* è la stessa di un sistema *Linux*, convenzionale, quindi adattare applicazioni *Linux* al sistema *μClinux* è un passo praticamente immediato. La distribuzione di *μClinux* inoltre è già predisposta per girare sui processori *Motorola MCF5272*, fatto che riduce ancora il numero di modifiche al codice sorgente che è stato necessario effetuare.

La programmazione per questo sistema operativo è realizzata tipicamente in linguaggio *C* o *C++*, anche se esistono implementazioni di macchine virtuali, che permettono lo sviluppo di applicazioni scritte in *Java*. Queste ultime risultano però più lente delle analoghe scritte in *C* e non verranno prese in considerazione in questa tesi.

[2] *Application Programming Interface*: In questo caso ci riferiamo all'insieme di chiamate al sistema operativo attraverso le quali si accede ai servizi offerti dallo stesso.

Tuttavia, anche questo sistema operativo presenta i suoi inconvenienti. La sua principale carenza è identificabile nella gestione di applicazioni grafiche e di interfacce utente grafiche, dal momento che il loro sviluppo può risultare più complesso se comparato con altri sistemi operativi, come ad esempio *Windows CE*. Data la finalità del sistema *Antares* questo inconveniente è tollerabile, soprattutto se proporzionato ai molteplici vantaggi offerti.

Tutte le informazioni sulla distribuzione *μClinux* possono essere ottenuti sulla pagina internet ufficiale [10].

1.8 Struttura della distribuzione μClinux

La distribuzione *μClinux* è nata per colmare la mancanza del sistema operativo *Linux* nel mercato dei sistemi embedded. Questa assenza era giustificata dal fatto che *Linux* è un sistema operativo avanzato che fa uso dei molteplici vantaggi offerti dalla unità di gestione della memoria (*MMU*), tipicamente è assente nella gran maggioranza dei microcontrollori orientati a sistemi embedded. Questa particolarità ha reso molto difficile il procedimento di adattare il *kernel* di *Linux* a questo tipo di microcontrollori, in quanto, compiti molto semplici in presenza di microprocessori *MMU*, come ad esempio la protezione delle zone di memoria di ciascun processo, diventano invece compiti complessi con microcontrollori non dotati di *MMU*.

Il primo microcontrollore sul quale è stato eseguito un *μClinux* è il *Motorola MC68328* . Oggigiorno esistono "port" di questo sistema per un gran numero di microprocessori, tra i quali vi si trovano anche i microcontrollori della famiglia *Coldfire*.

La distribuzione *μClinux* è formata da 3 blocchi di *software* principali:

- *il kernel:* in questa distribuzione sono inclusi i "port" del *kernel* di *Linux*. della versione *2.2.x* e della *2.4.x* . Verrà usata la versione più recente.
- *la biblioteca standard C (libc):* si trata di due implementazioni della biblioteca standard del C per microcontroller, la *uC-libc* e la *uClibc*. la differenza tra queste due, pur essendo minima, é dovuta al fatto che la *uC-libc* sebbene sia completa, non segue rigorosamente lo standard, per facilitare la migrazione di applicazioni specifiche per microcontrollori embedded. Di conseguenza è racco-

mandabile l'uso della biblioteca *uClibc*, più standard e completa, a meno che qualche applicazione non richieda specificamente la scelta della *uC-libc*

- *le applicazioni:* nella distribuzione *μClinux* sono incluse molteplici applicazioni, sia di sistema (come ad esempio l'interprete dei comandi), sia applicazioni per l'utente (server *Web*).

Una volta effettuato il processo di compilazione della distribuzione, viene generato un file binario unico contenente il *kernel* e il "file system" tipico di un qualsiasi sistema *Linux*, dove si possono trovare le applicazioni, le librerie ed i file di configurazione del sistema.

Capitolo 2

Il programma Ed68k

Il programma *Ed68k* nasce dall'esigenza di offrire agli studenti del corso di *Sistemi Elettronici Digitali* un'interfaccia grafica amichevole, che permetta di interagire con semplicità con la scheda *Antares* e con il firmware *dBUG* descritto nella sezione 4.

2.1 Il laboratorio LSED

In questo corso, denominato *Laboratorio di Sistemi Elettronici Digitali*, gli studenti devono progettare e costruire alcuni circuiti da connettere alle porte di ingresso/uscita della scheda *Antares* e programmare la scheda stessa, in linguaggio *Assembler* o *C*, al fine di ottenere il funzionamento richiesto dal tema dell'esercitazione. Solitamente si tratta di creare un sistema di elaborazione numerica dei segnali, che permetta di:

- amplificare e attenuare
- ritardare
- filtrare, potendo variare i coefficienti di una funzione di trasferimento assegnata.

Per fare ciò, sarà necessario connettere alle porte di ingresso/uscita dei convertitori analogico/digitale e digitale/analogico, che saranno pilotati dalla stessa scheda *Antares* tramite appositi segnali.

L'esercitazione potrà poi essere arricchita aggiungendo componenti e funzionalità al sistema. Ecco alcune delle estensioni possibli.

- Visualizzare dati, relativi al processamento in esecuzione, su di un display *LCD* a 7 segmenti.
- Effettuare il *controllo remoto* del programma, con un comando ad infrarossi o a radiofrequenza.
- Effettuare il *controllo remoto* del programma, con l'ausilio di una calcolatrice HP48, HP49, di un calcolatore palmare o di un portatile.
- Aggiungere funzionalità di *riconoscimento vocale*.
- Aggiungere funzionalità di *sintesi vocale*.
- Implementare un semplice *frequenzimetro digitale*.
- Implementare un semplice generatore di funzioni programmabile.

2.2 Funzionamento del programma Ed68k

Il programma *Ed68k* svolge pertanto il ruolo di interfaccia utente, permettendo di effettuare tutte le operazioni richieste senza dover conoscere esattamente l'hardware e le funzionalità della scheda *Antares* e senza doversi preoccupare della complicata sintassi necessaria per pilotare il compilatore e il firmware "dBUG".

Ecco i compiti principali del programma *Ed68k*.

- Fornire un *editor* dove sia possibile scrivere codice in C o in Assembler.
- *Compilare* il codice editato e segnalare gli errori. Questa parte sarà approfondita nella sezione 2.3.4.
- *Inviare* alla scheda i file binari compilati.
- Permettere il *debug* del software, tramite i comuni tool di esecuzione *passo a passo*, *watch* di variabili e registri, *breakpoint* .... Questa parte sarà approfondita nella sezione 2.3.5 e seguenti.

- Presentare un *terminale* per ricevere e inviare dati al software in esecuzione sulla scheda, tramite la connessione seriale.
- Gestire le principali situazioni di *errore* che si possono verificare o intercettare la pressione del tasto *RESET* posto sulla scheda.
- Modificare in maniera agile i *registri* del processore.
- Modificare la *memoria* della scheda e il contenuto delle *variabili*.

Appena avviato, il programma *Ed68k* si presenta come visualizzato nella ***Figura 2.1***

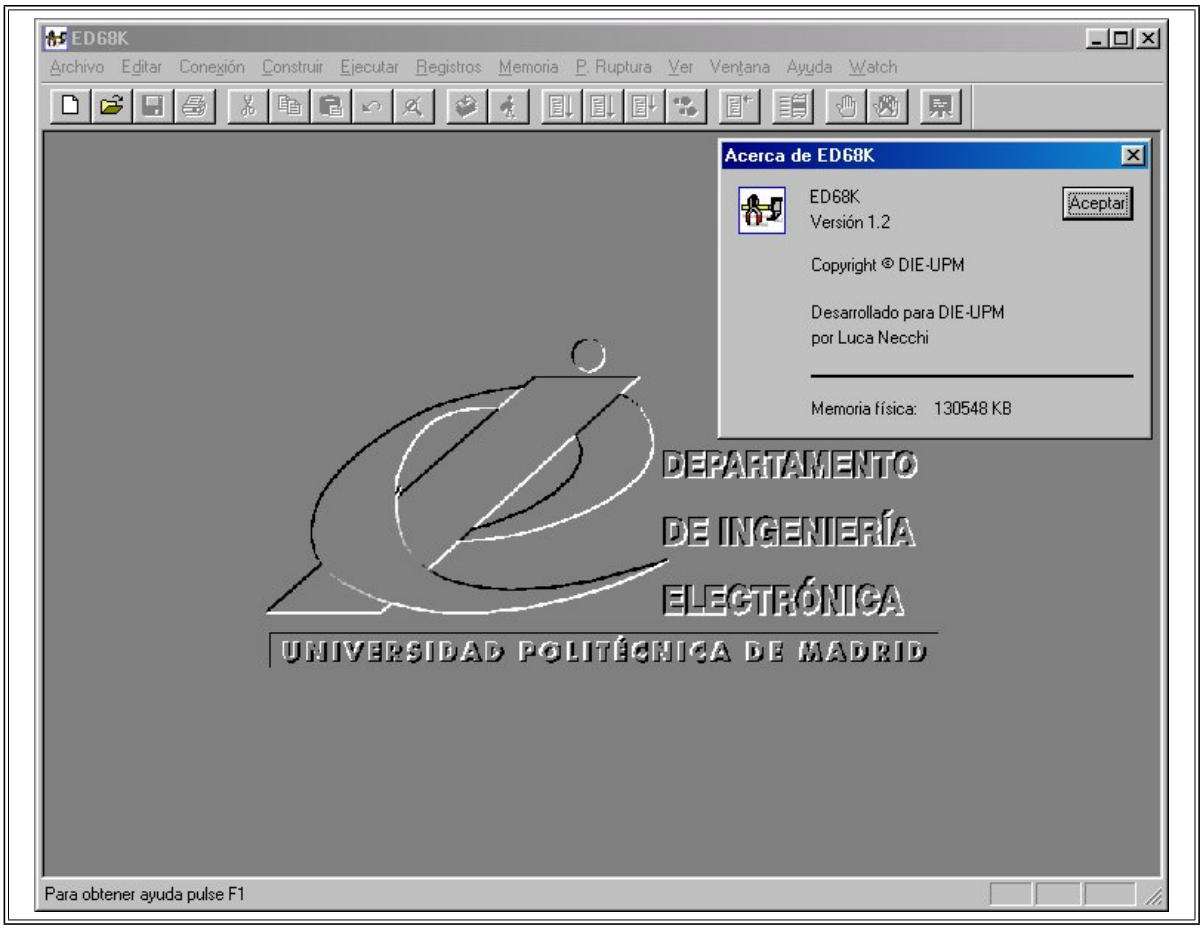


Figura 2.1: *Schermata iniziale del programma Ed68k*

2.3 Descrizione delle funzioni del programma Ed68k

Descriviamo tutte le funzioni del programma *Ed68k* analizzando i comandi messi a disposizione nei vari menu.

2.3.1 File

In questo menu vi sono le funzioni di accesso ai file, tipiche di qualsiasi programma Windows[1].

Figura 2.2: *Menu Archivo*

[1] L'uso dello spagnolo nei menù di questo programma è dovuto al fatto che il software è stato sviluppato per l'Università Politecnica de Madrid

2.3.2 Modifica

In questo menu vi sono i comandi di gestione dell'editor, quali taglia, copia e incolla.

Figura 2.3: *Menu Editar*

2.3.3 Connessione

In questo menu vi sono i comandi per gestire la connessione seriale con la scheda *Antares*.

Figura 2.4: *Menù Conexiòn*

- **Abrir**: apre la porta seriale, verifica l'effettivo funzionamento della connessione con la scheda *Antares* e aggiorna la tabella di visualizzazione dei registri.
- **Cerrar**: chiude la connessione seriale.
- **Configurar**: configurazione della porta seriale. Permette di scegliere la porta seriale da utilizzare e la velocità di comunicazione. Qualora venga cambiata la velocità di connessione, il programma provvederà anche ad inviare il comando di cambio velocità alla scheda *Antares* e il comando di *Reset*. Questo comando andrà a buon fine solo se la velocità del *PC* e della scheda *Antares* sono uguali prima del cambio.

Figura 2.5: *Configurazione porta seriale*

2.3.4 Compilazione

In questo menu si trovano i comandi necessari per compilare programmi in linguaggio *Assembler* e *C*. I menu saranno selezionabili solamente se disponibile nell'editor il relativo programma *Assembler* o *C*.

Figura 2.6: *Menù Construir*

- **Lugar en RAM**: permette di definire la posizione di memoria nella quale risiederà il programma (Text) e nella quale risiederanno le variabili (Data). Questi parametri verranno passati al *linker* e saranno memorizzati nel file mem.ini, in modo da non doverli digitare nuovamente ad ogni esecuzione del programma *ed68k*. È compito dello studente inserire in questi campi dei valori adatti, che rientrino nell'area di memoria *RAM* e che non causino la sovrapposizione dei relativi blocchi di memoria.

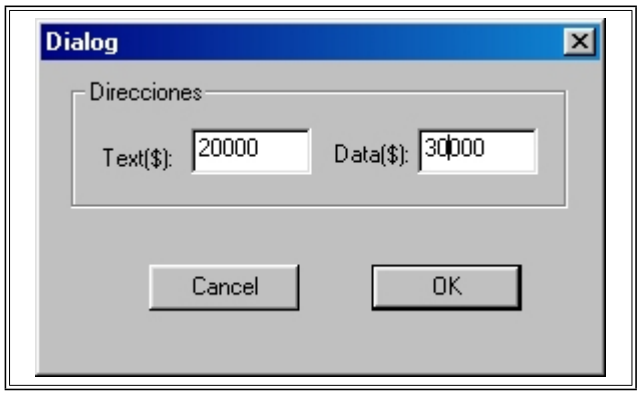


Figura 2.7: *Configurazione parametri del linker*

- **Ensemblar**: effettua la chiamata all'assemblatore, nel nostro caso il programma *AS* della *GNU*, per verificare la correttezza del codice scritto dall'utente.

Verrà riportata, in una nuova finestra, la lista e il conteggio degli eventuali *errori* e *warning*. Questo comando non produce nessun file eseguibile come output.

- **Make AS**: effettua la chiamata all'assemblatore e al linker, tramite l'ausilio di un file batch, effettuando così l'intero processo di assemblaggio, restituendo un file in formato s-record (con estensione .h68) pronto per essere caricato, tramite la connessione seriale, nella memoria della scheda *Antares*.

- **Compilar**: effettua la chiamata al compilatore, nel nostro caso il programma *GCC* della *GNU*, per verificare la correttezza del codice scritto dall'utente e riporta in una nuova finestra la lista e il conteggio degli eventuali *errori* e *warning*. Questo comando non produce nessun file eseguibile come output.

- **Make GCC**: effettua la chiamata al precompilatore, al compilatore, all'assemblatore e al linker, tramite l'ausilio di un file batch, effettuando così l'intero processo compilazione. Durante questa fase verranno anche compilati altri file, quali la routine di start e le routine di gestione degli interrupt e verrà restituito un file in formato s-record (con estensione .h68) pronto per essere caricato, tramite la connessione seriale, nella memoria della scheda *Antares*.

- **Cargar**: per inviare un file, tramite la connessione seriale, alla memoria della scheda *Antares*. Verrà chiesto quale file caricare, verrà aperta la finestra di debug con il listato del programma, verrà evidenziata la prima linea eseguibile del programma (entry point), verranno impostati i registri PC[2] e SR[3]. Il programma risulterà pronto per essere eseguito.

[2]Program Counter
[3]Status Register

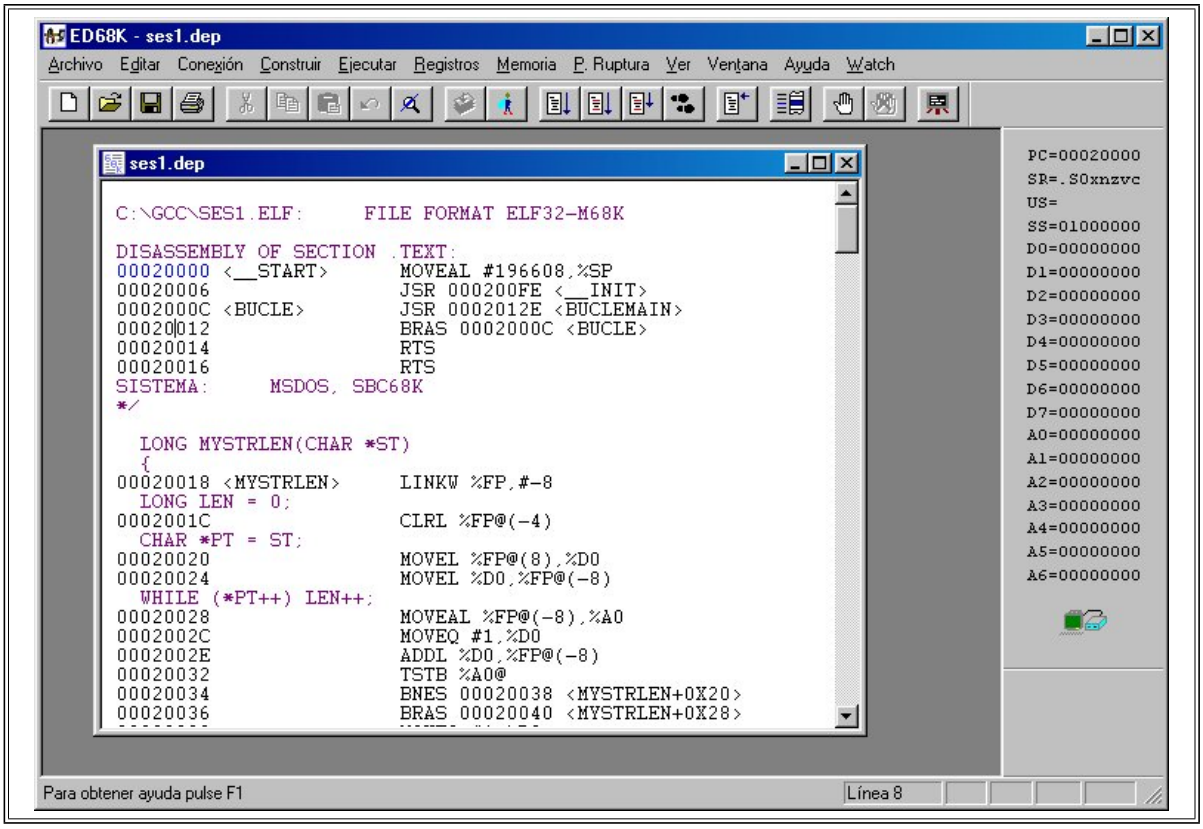


Figura 2.8: *Programma caricato sulla scheda*

2.3.5 Esecuzione

In questo menu si trovano i comandi necessari per l'esecuzione e il debug del programma precedentemente caricato nella memoria della scheda *Antares*.

Figura 2.9: *Menù Ejecutar*

- **Sin P. Ruptura:** esegue il programma senza badare a breakpoint e senza poter visualizzare il valore di registri o variabili. Sarà possibile interrompere l'esecuzione del programma solamente premendo il tasto *Reset* posto sulla scheda *Antares*.

Figura 2.10: *Evento Reset*

- **Con P. Ruptura:** esegue il programma fino all'istruzione precedente a quella dove vi è un *breakpoint*. La posizione di memoria dell'istruzione successiva sarà colorata in blu e saranno aggiornati i valori dei registri e delle variabili presentate nella finestra di *watch*.
- **Hasta el cursor:** esegue il programma fino all'istruzione precedente a quella dove si trova il cursore. La posizione di memoria dell'istruzione successiva sarà colorata in blu e saranno aggiornati i valori dei registri e delle variabili presentate nella finestra di *watch*.
- **Paso a paso:** esegue una singola istruzione. La posizione di memoria dell'istruzione successiva sarà colorata in blu e saranno aggiornati i valori dei registri e delle variabili presentate nella finestra di *watch*.

2.3.6 Registri:

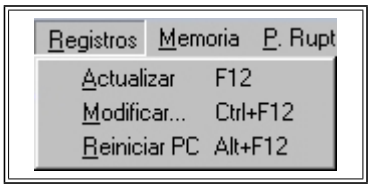


Figura 2.11: *Menù Registros*

- **Actualizar**: aggiorna tutti i valori presentati nella finestra dei registri. Questo comando viene automaticamente invocato da tutti i comandi che possono causare cambi nel valore delle variabili, come ad esempio il comando *passo-passo*, in modo da mantenere costantemente aggiornata la lista di valori quando si effettua il debug del codice.
- **Modificar**: permette di modificare il valore dei registri del processore, tramite la finestra visualzzata nella ***Figura 2.12***.

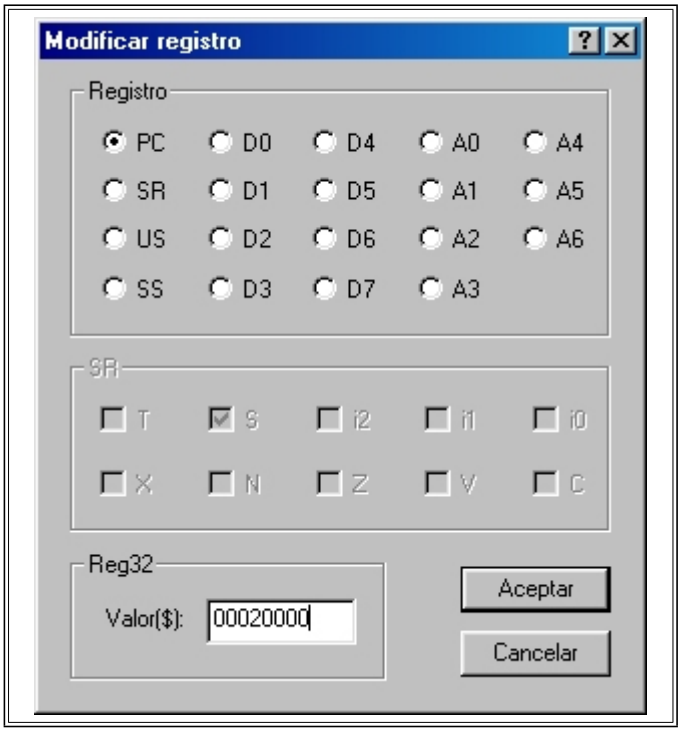


Figura 2.12: *Modifica Registri*

- **Reiniciar PC**: assegna al registro PC[4] il valore iniziale, ovvero salta all' "entry point" del programma.

[4]Program Counter

2.3.7 Memoria:

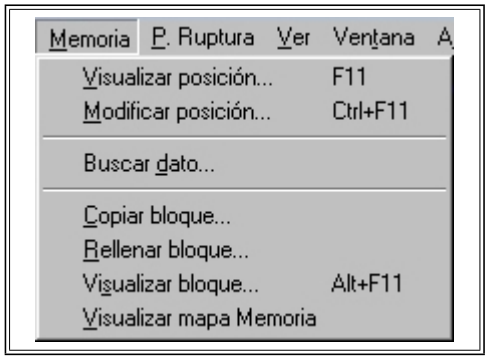


Figura 2.13: *Menù Memoria*

- **Visualizar posiciòn:** permette di visualizzare il contenuto esadecimale di una variabile o di una qualsiasi posizione di memoria della scheda. Sarà possibile specificare l'indirizzo esadecimale della posizione di memoria, e la dimensione della relativa variabile o immetterne direttamente l'etichetta. I parametri mancanti, in questo caso, saranno riempiti premendo il tasto "Busca direccion".

Figura 2.14: *Visualizzare posizione di memoria*

- **Modificar posiciòn:** permette di modificare il contenuto di una variabile o di una qualsiasi posizione di memoria che non sia di sola lettura. Sarà possibile specificare l'indirizzo esadecimale della posizione di memoria e la dimensio-

ne della relativa variabile o immetterne direttamente l'etichetta. I parametri mancanti, in questo caso, saranno riempiti premendo il tasto "Busca direccion".

Figura 2.15: *Modificare posizione di memoria*

- **Buscar dato:** permette di cercare un dato in un blocco di memoria specificato.

Figura 2.16: *Cerca dato*

- **Copiar bloque:** permette di copiare un blocco di memoria in un'altra posizione.

Figura 2.17: *Copiare blocco di memoria*

- **Rellenar bloque:** permette di riempire un blocco di memoria con un dato.

Figura 2.18: *Riempire blocco di memoria*

- **Visualizar bloque:** permette di visualizzare il contenuto di un intero blocco di memoria. L'operazione è limitata a una quantità di memoria di 16K.

Figura 2.19: *Visualizza blocco di memoria*

- **Visualizar mapa de memoria:** visualizza la mappa di memoria della scheda *Antares*, specificando gli indirizzi verso i quali sono mappate le varie zone di memoria e le linee di chip-select relative.

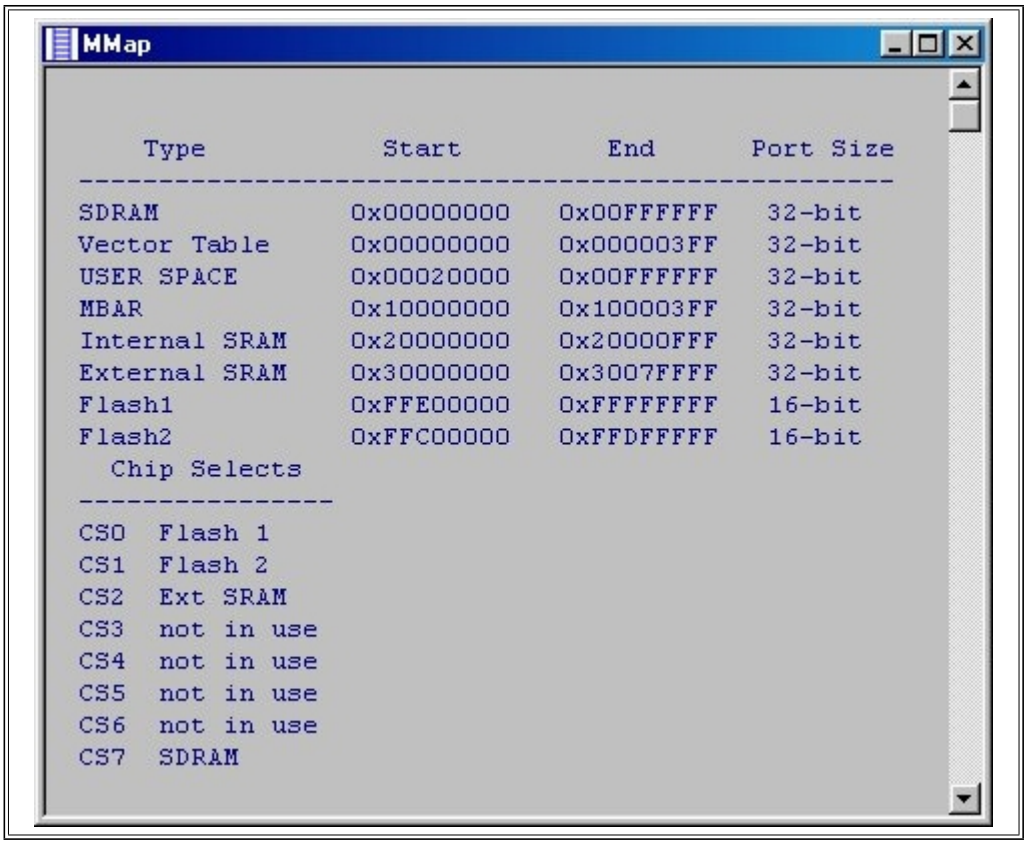


Figura 2.20: *Visualizza mappa di memoria*

2.3.8 Breakpoint

In questo menu si trovano i comandi necessari per la gestione dei *breakpoint*.

Figura 2.21: *Menù P.Ruptura*

- **Insertar.../Eliminar:** verrà aggiunto (o rimosso, nel caso già esista) un *breakpoint*, nella posizione indicata dal cursore. Sarà possibile assegnare un valore ad un contatore, affinchè l'esecuzione venga interrotta solamente dopo un certo numero di passaggi alla posizione indicata. Il *breakpoint* verrà indicato nel listato colorando l'indirizzo della posizione in questione di rosso.

Figura 2.22: *Aggiunta di un breakpoint*

- **Eliminar Todos:** rimuove tutti i *breakpoint* precedentemente impostati.
- **Visualizar:** presenta, in un apposita finestra: lista di tutte le posizioni di *breakpoint* (elencate sotto la voce "Address"), il contatore del numero di passaggi già effettuati a tale posizione (elencato sotto la voce "count") e il "trigger". L'interruzione dell'esecuzione avverrà solamente quando il contatore sarà multiplo esatto del valore di trigger.

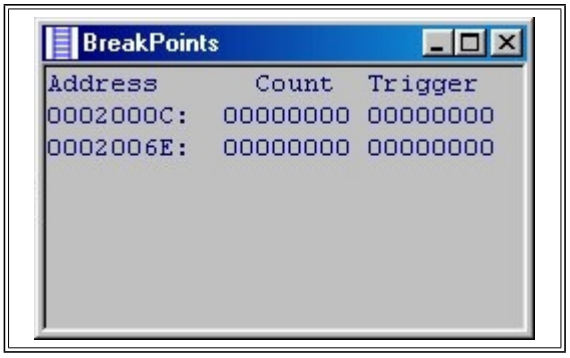


Figura 2.23: *Visualizzare la lista di breakpoint*

2.3.9 Visualizza

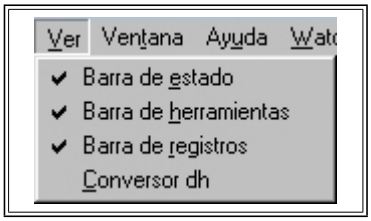


Figura 2.24: *Menù Ver*

- **Barra de estado:** mostra o nasconde la barra di stato, posta nella parte bassa dello schermo.

Figura 2.25: *Barra di stato*

- **Barra de hierramentas:** mostra o nasconde la barra con i tools più usati, posta immediatamente sotto ai menu.

Figura 2.26: *Barra di tools*

- **Barra de registros:** mostra o nasconde la barra dei registri, posta nella parte destra dello schermo.

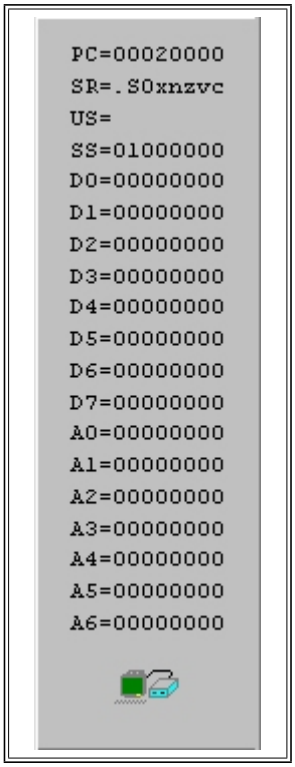


Figura 2.27: *Barra dei registri*

- **Conversor dh:** effettua la conversione tra numeri binari, decimali ed esadecimali.

Figura 2.28: *Convertitore Binario, Decimale, Esadecimale*

2.3.10 Finestra

Comandi per la gestione delle finestre aperte.

Figura 2.29: *Menù Ventana*

- **Cascada:** ordina le finestre aperte in modo che siano di dimensioni identiche e che si sovrappongano in maniera ordinata, in una sorta di cascata.
- **Mosaico horizontal:** divide orizzontalmente lo spazio disponibile tra le varie finestre.
- **Mosaico vertical:** divide verticalmente lo spazio disponibile tra le varie finestre.
- **Terminal:** mostra o nasconde la finestra “terminale” da dove sarà possibile comunicare con il programma in esecuzione sulla scheda *Antares*.

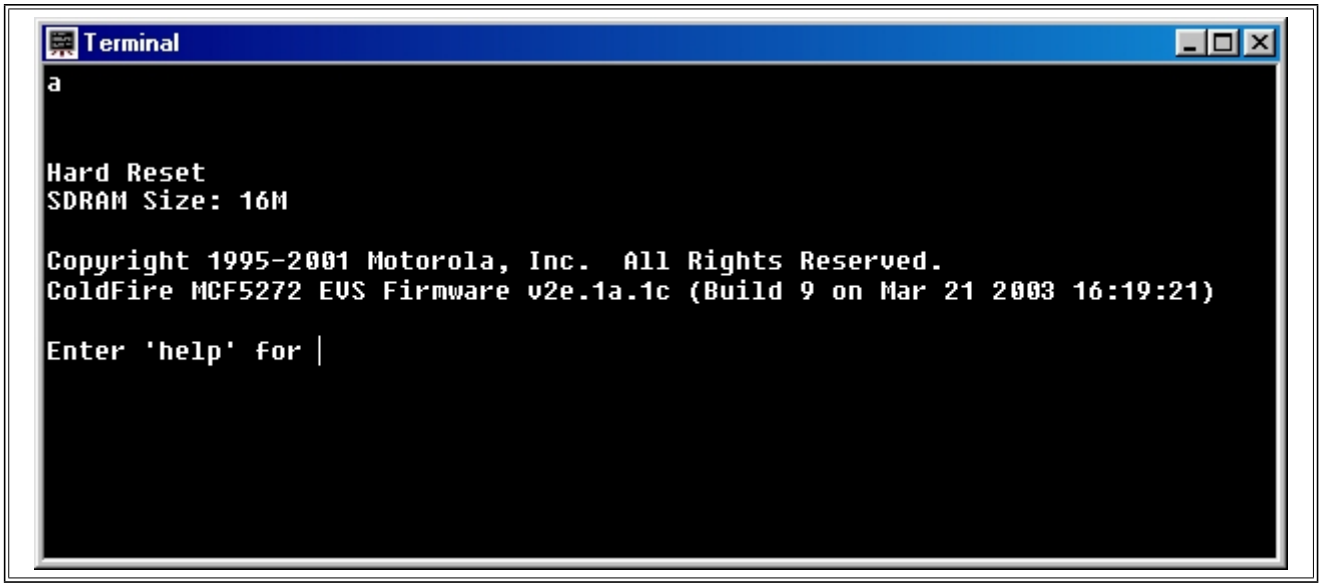


Figura 2.30: *Terminale*

- **Watch:** mostra o nasconde la finestra “watch” descritta nella sezione 2.3.12.
- **Limpiar terminal:** cancella il contenuto della finestra “terminale”.

2.3.11 Help

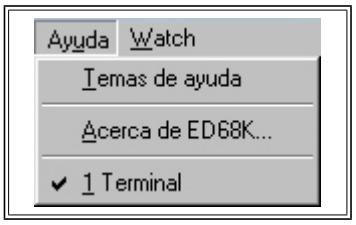


Figura 2.31: *Menù Ayuda*

- **Temas de ayuda:** apre la guida in linea.
- **Acerca de ED68K:** presenta versione, copyright e autore del programma *Ed68k*.

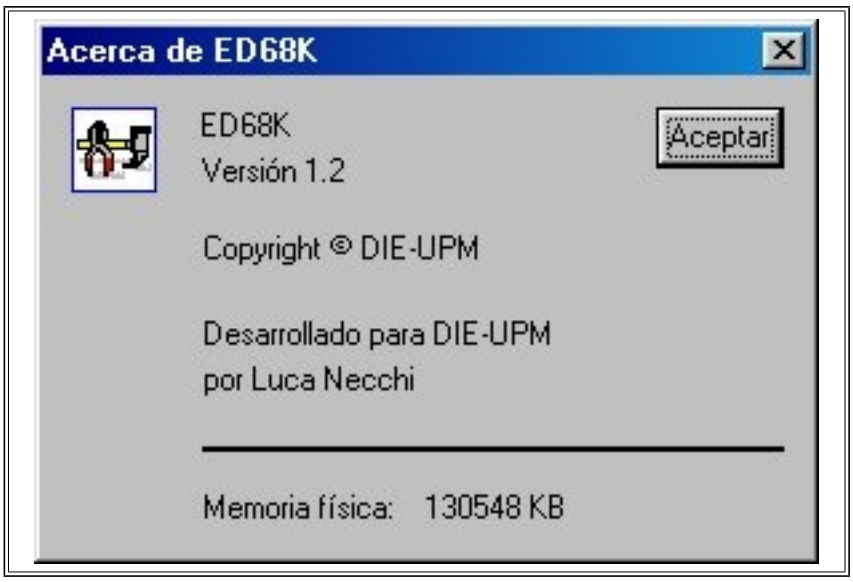


Figura 2.32: *Finestra "Acerca"*

2.3.12 Watch:

Figura 2.33: *Menù Watch*

- **Watch**: apre la finestra di Watch delle variabili, visibile in ***Figura 2.34***.

Figura 2.34: *Finestra di Watch*

Una volta selezionata una variabile, indicando la sua etichetta o la sua posizione di memoria, è possibile aggiungerla o rimuoverla dall'elenco di variabili della finestra di *watch* illustrata in ***Figura 2.35***.

- **Refresh**: aggiorna tutti i valori presentati nella finestra di watch. Questa funzionalità viene automaticamente chiamata da tutti i comandi che possono causare cambi nel valore delle variabili, come ad esempio il comando *passo-passo*, in modo da mantenere costantemente aggiornata la lista di valori quando si effettua il debug del codice.

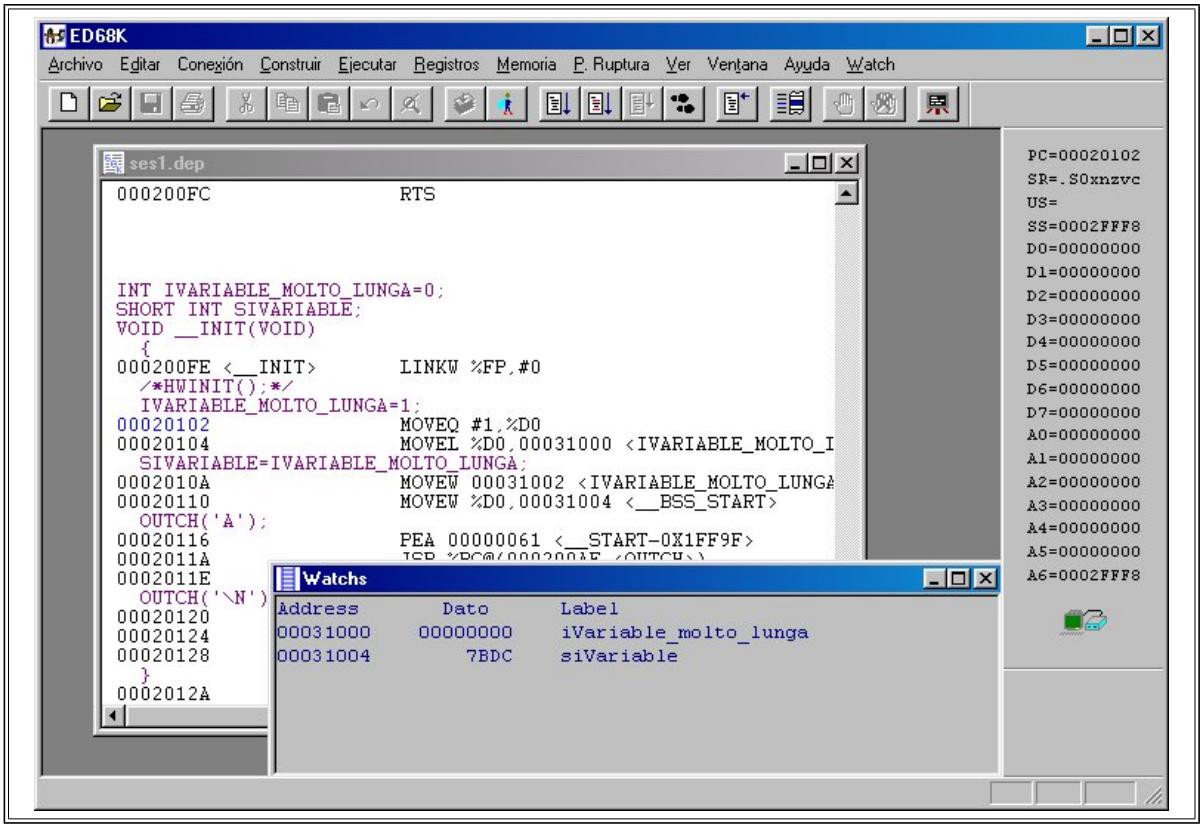


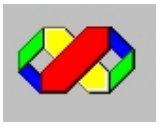
Figura 2.35: *Finestra di Watch*

2.4 Programmare in C

Il programma *Ed68k*, come già visto, permette di editare e compilare codice scritto in linguaggio *Assembler* o *C*. Di contro, le funzionalità di debug offerte dal firmware "dBUG" sono orientate all'esecuzione di istruzioni in linguaggio macchina, per loro stessa natura direttamente correlate a istruzioni *assembler*. È stato pertanto deciso di effettuare il debug di codice scritto in *C* presentando un listato "ibrido". Ogni istruzione in linguaggio *C*, evidenziata in colore viola, verrà seguita da istruzioni *Assembler*. Le istruzioni *Assembler* saranno quelle realmente eseguite, evidenziate dall'esecuzione *passo a passo*, mentre le istruzioni *C* avranno un significato puramente mnemonico per l'utente. Questa peculiarità è mostrata dalla ***Figura 2.8***.

Capitolo 3

Struttura del programma Ed68k



Il programma *Ed68k* è stato scritto principalmente in *C++* con Microsoft *Visual Studio 6* e fa largo uso delle classi e librerie grafiche proprietarie messe a disposizione da questo tool. Il programma è completato da alcuni file batch e da una versione del noto compilatore della *GNU* modificata per produrre file binari adatti a lavorare sulla scheda *Antares*.

3.1 La scelta del compilatore

Per poter eseguire programmi sulla piattaforma *Antares* occorre che questi siano stati compilati usando un compilatore specifico. In particolare il compilatore deve presentare le seguenti caratteristiche:

- deve produrre file binari per un processore *MCF5272* della famiglia *Motorola Coldfire*, processore che è stato ampiamente descritto nella sezione A.1.
- deve funzionare in un ambiente Windows, in quanto il programma *ed68k* lavora in tale ambiente.
- deve essere utilizzabile tramite “linea di comando”.
- deve essere gratuito o molto economico, in quanto una copia di tale compi-

latore deve essere installata su ogni computer del laboratorio dove verranno impartite le esercitazioni del corso *LSED*[1].

Analizzati tali requisiti, è stato scelto di utilizzare il noto compilatore della *GNU*. Nel seguito viene descritto questo compilatore e viene spiegato come poterlo installare in un ambiente *Windows*. Nel seguito, indicheremo questo compilatore con il nome "gcc-m68k" per non confonderlo con il compilatore di sistema.

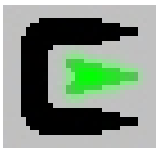
3.2 Il compilatore della GNU.

In una qualsiasi distribuzione di *Linux* è possibile trovare il compilatore *GCC*, ovvero il Gnu C Compiler, l'assemblatore *AS*, il linker *LD*. Questi programmi fanno parte dei due pacchetti *gcc* e *binutils* entrambi distribuiti sotto licenza *GPL*, il che vuol dire che il codice ivi contenuto può essere liberamente distribuito, utilizzato e modificato, a patto di ridistribuire nuovamente il codice sotto licenza *GPL*.

Una licenza di questo tipo rende possibile eseguire detto codice su qualsiasi architettura di sistema e qualsiasi sistema operativo, in quanto la presenza del codice sorgente permette di adattare totalmente il programma alle nostre esigenze. Le operazioni da svolgere nel nostro caso specifico, per adattare il compilatore, sono le seguenti:

- modificare, tramite l'uso di alcune patch, il codice sorgente del compilatore, affinchè supporti l'architettura *ColdFire*.
- compilare tale codice per un ambiente Windows.

3.3 L'ambiente CYGWIN



Cygwin è un ambiente di lavoro *Unix*, che si installa su PC con sistema operativo Windows. È possibile reperire questo software all'indirizzo internet [11]. La procedura di installazione di *Cygwin* è molto semplice. Sarà sufficiente scaricare dalla rete il programma di *setup* e, grazie ad

[1] "Laboratorio de Sistemas Electrònicos Digitales"

un interfaccia grafica, selezionare i pacchetti desiderati. Il setup stesso provvederà in seguito a scaricare, dal mirror selezionato, i soli pacchetti scelti. Per le nostre esigenze sarà necessario aggiungere alla configurazione di default i pacchetti “devel” dove vi sono il compilatore, l’assemblatore e il programma “ patch”.

Una volta installato *Cygwin* sul proprio *PC*, potremo utilizzare una console “bash” ben nota a qualsiasi utente *Linux* e da questa effettuare tutte le operazioni tipiche di una console *Linux*, nonostante il sistema operativo del PC sia Windows.

Il pacchetto *Cygwin* permette inoltre di installare molteplici tool grafici, come il server *XFree*, e di conseguenza utilizzare applicativi grafici tipici di un sistema *Linux*. Per i nostri scopi non sarà necessaria nessuna di queste funzionalità.

È importante notare che il sistema *Cygwin* funziona grazie ad alcune librerie che sono memorizzate nella sottodirectory */BIN* nel path dove abbiamo installato *Cygwin* e che vengono aggiunte alla variabile “PATH” di sistema solamente durante l’esecuzione di *Cygwin*. Aggiungendo questa directory al path di sistema saremo in grado di utiizzare gli applicativi *unix* offerti da *Cygwin* anche senza lanciarli dalla console “bash”. Questo risulterà molto importante per noi, in quanto sarà lo stesso programma *Ed68k* a dover effettuare chiamate al compilatore.

3.4 Le modifiche al compilatore “gcc-m68k”

Vediamo ora quali sono le patch che è stato necessario usare.

Patch per il pacchetto “binutils”

- binutils-2.10-coldfire-fixes.diff
- binutils-2.10-coldfire-cpushl.diff

Patch per il pacchetto “gcc”

- gcc-2.95.3-coldfire-epilogue-fix.diff
- gcc-2.95.3-coldfire-hwdiv.diff
- gcc-2.95.3-coldfire-longmul.diff
- gcc-2.95.3-interrupt-attribute-20010127.diff

- gcc-2.95.3-coldfire-version-message.diff
- gcc-2.95.3-coldfire-alignment.diff

Patch per il pacchetto “newlib”

- newlib-1.9.0-coldfire-20010318.diff

come sommariamente indicato dal nome di ogni file, queste patch modificano alcuni dettagli del codice sorgente del compilatore, adattandolo alle nostre esigenze.

3.5 Compilazione

Il programma *Ed68k*, come fornito nel *CDROM* allegato, dispone di una versione di questo compilatore, che è stato configurato su un computer dotato di processore *Athlon*. Trattandosi di file binari ottimizzati per questa precisa architettura, risulterà necessario rieffettuarne la compilazione, qualora il programma venga installato su un computer con architettura differente.

3.5.1 Installazione del compilatore gcc-m68k

Per fare ciò, avviamo una console *Cygwin*.

Usando il pacchetto “`gcc-m68k-source-r7-20010415-gnu-source`” che si trova nel *CDROM* allegato, e liberamnte scaricabile all’indirizzo internet [16], sarà possibile automatizzare il processo di compilazione, effettuando in una volta sola tutti i passi necessari. Sarà necessario posizionarsi nella cartella contenente il file“`gcc-m68k-source-r7-20010415-gnu-source.tar.bz2`” e scompattarlo con il comando

```
# tar -xvjf gcc-m68k-source-r7-20010415-gnu-source.tar.bz2
```

e avviare lo script di configurazione[2], dopo essersi spostati nella directory appena creata, con i comandi

[2]Se si utilizza un computer dotato di Windows NT o 2000, sarà necessario accertarsi che la console *Cygwin* venga eseguita con i diritti di amministratore di sistema.

```
# cd gcc-m68k-source-r7-20010415-gnu-source
# ./build-gcc.sh /gcc-m68k/
```

Il parametro `/gcc-m68k` specifica allo script in quale directory installare i file binari. Per conoscere esattamente le modalità di uso dello script sopra citato, si rimanda al file `readme` contenuto nella directory:
`gcc-m68k-source-r7-20010415-gnu-source`

Lo script `build-gcc` provvederà inoltre a creare, nella directory:
`gcc-m68k-source-r7-20010415-gnu-source/build/package` 3 pacchetti compressi, ovvero:

- gcc-m68k-win32-r7-20010415.tar.bz2
- cpu32libs-win32-r7-20010415.tar.bz2
- extralibs-win32-r7-20010415.tar.bz2

che contengono, rispettivamente, il compilatore per i processori della sola famiglia coldfire e le librerie aggiuntive per le restanti famiglie, a 32 bit e non. Vale a dire che, estraendo i 3 pacchetti, otterremo un compilatore assolutamente generico, mentre, estraendo solo il primo, otterremo il solo compilatore per i processori della famiglia *coldfire*. La nostra scelta, sarà quella di utilizzare il solo compilatore specifico, che installeremo, usando *Cygwin*, con i seguenti comandi:

```
# cd /cygdrive/c/ed68k
# tar -xvjf gcc-m68k-source-r7-20010415-gnu-source/build
  /package/gcc-m68k-win32-r7-20010415.tar.bz2
```

3.5.2 Aggiornamento delle librerie di Cigwin

Qualora venga aggiornato il compilatore "gcc-m68k", sarà necessario copiare nuovamente le librerie di *cygwin*. Per fare questo, daremo il seguente comando ad una console *Cygwin*:

```
# cp /bin/cyg*.dll /cygdrive/c/ed68k/gcc-m68k/system
```

Il path di queste librerie verrà aggiunto al path di sistema all'atto di avviare il programma *Ed68k*.

Capitolo 4

Il firmware dBUG

4.1 Cosa è il dBUG

Il dBUG è un firmware prodotto da *Motorola* e destinato a molte delle schede dimostrative su cui sono installati i processori *Motorola*

Questo firmware offre un'interfaccia testuale che può essere usata per scaricare, eseguire codice e gestire le funzioni basilari di un ambiente di "debug".

L'interfaccia utente è di tipo "linea di comando" e per il suo utilizzo si dovrà utilizzare un terminale seriale.

L'attuale versione del dBUG supporta le seguenti famiglie di microcontrollori:

- M68000
- ColdFire MCF5200
- PowerPC

Il codice sorgente di questo firmware viene distribuito liberamente sul sito internet [13] all'indirizzo:
http://e-www.motorola.com/collateral/DBUGFIRM.html

Detto codice è protetto da una specifica licenza *Motorola* e, nei termini della stessa, può essere modificato e ridistribuito. Vengono inoltre distribuite dettagliate istruzioni per l'utilizzo del *dBUG* nel manuale [5], reperibile sul sito internet [13]

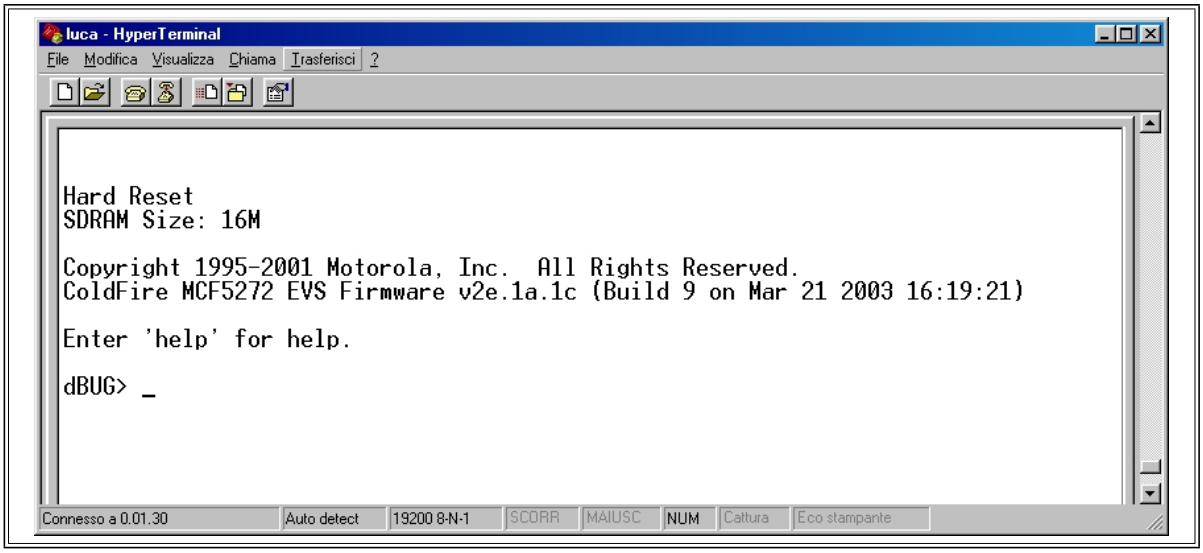


Figura 4.1: *Prompt del firmware dBUG*

4.2 Struttura del dBUG.

Il codice sorgente del firmware *dBUG* viene distribuito su file separati, per permettere all'utente di scaricare da internet i soli file strettamnte necessari. nel nostro caso è stato necessario procurarsi:

- DBUGDSS.tgz
- DBUGM5272C3.tgz

Nel file DBUGDSS.tgz troveremo il codice sorgente generico per qualsiasi architettura, mentre nel file DBUGM5272C3.tgz troveremo tutto ciò che riguarda la scheda dimostrativa *Motorola MCF5272C3*.

Sarà quindi necessario, partendo da questa struttura di base, modificare il dbug per ottenerne il funzionamento sulla scheda *Antares*, che, come già detto, presenta caratteristiche differenti dalla scheda dimostrativa *Motorola MCF5272C3*.

4.3 Modifiche al dBUG

In questa sezione analizzeremo le modifiche che è stato necessario effettuare per ottenere il funzionamento del *dbug* sulla scheda *Antares*.

Le principali differenze tra la scheda *Antares* e la scheda dimostrativa *Motorola MCF5272C3* sono:

- **memoria Flash:** la scheda *Motorola* è provvista di un solo chip da *2 MBytes* di *Flash* del produttore *AMD* [1], mentre *Antares* è provvista di due chip da *2 MByte* ciascuno, del produttore *Fujitsu* [3], è stato pertanto necessario modificare il driver della *Flash*.
- **memoria SDRAM:**la scheda *Motorola* dispone di *8 MBytes*, mentre *Antares* è provvista di *16 Mbytes* de memoria *SDRAM*.
- **ingressi/uscite:** la scheda *Antares* è dotata di un insieme di porte di ingresso e uscita dati, quali le due porte da 8 bit digitali, o i *LED* generici.

Sono state inoltre aggiunte alcune funzioni per adattare il firmware dBUG a precise necessità dettate dall'uso che sarà fatto della scheda *Antares* all'interno del *DIE* - "Dipartimento di Ingegneria Elettronica" dell'Università di Madrid:

- **UPFLASH**: Upload della primo chip di memoria /emphflash
- **UPFLASH2**: Upload della secondo chip di memoria /emphflash
- **COLILO**: Avvia l'esecuzione del *CoLiLo* e di conseguenza il *μClinux*

4.4 Utilizzo del dBUG

Lo scopo principale del programma *Ed68k*, descritto nel capitolo 2 è quello di offrire all'utente un'interfaccia comoda, interagendo in maniera trasparente con il firmware della scheda *Antares*, ovvero il *dBUG*. Quest'ultimo, infatti, offre un gran numero di comandi dedicati alla gestione della scheda e al debug dei programmi caricati in memoria.

Ecco la lista dei comandi accettati dal dBUG.

- **BF - Block Fill:** permette di riempire un blocco di memoria con un dato.
- **BM - Block Move:** permette di spostare un blocco di memoria.
- **BS - Block Search:** permette di cercare un dato in un blocco di memoria.

- **BR - Breakpoints:** permette di inserire e rimuovere breakpoint.
- **COLILO:** avvia il "coldfire Linux Loader", che provvederà all'esecuzione del μClinux.
- **DATA - Data Conversion:** permette di effettuare la conversione tra numeri esadecimali, decimali e binari.
- **DI - Disassemble:** permette di disassemblare, in maniera semplice, parti di codice contenute in memoria.
- **DL - Download Console:** permette alla scheda di ricevere un file tramite la connessione seriale.
- **DN - Download Network:** permette alla scheda di ricevere un file tramite la connessione ethernet.
- **GO - Execute:** esegue un programma caricato in memoria.
- **GT - Execute To:** esegue un programma, arrestandolo alla posizione di memoria specificata.
- **HELP - Help:** visualizza una schermata di aiuto.
- **MMAP - Display a memory map:** Visualizza la mappa completa della memoria e la configurazione delle linee di chip select. Tale mappa è visibile nella ***Figura 2.20***.

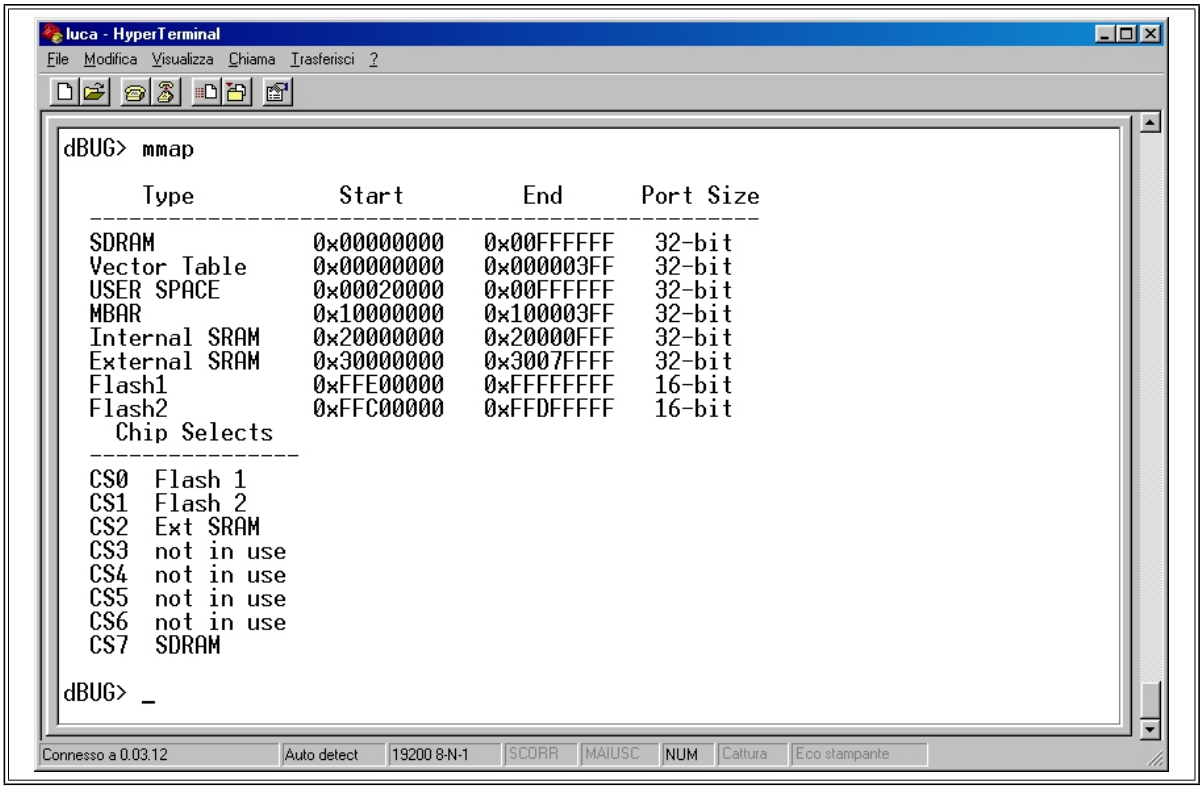


Figura 4.2: *Mappa di memoria*

- **MD - Memory Display:** visualizza il contenuto di ampie zone di memoria.
- **MM - Memory Modify:** modifica e visualizza il contenuto di piccole zone di memoria (al massimo 32 bit).
- **RD - Register Display:** visualizza il valore dei registri del processore.
- **RM - Register Modify:** modifica il valore dei registri del processore.
- **RESET - Reset the Board and dBUG:** Riavvia la scheda.
- **SET - Set Configurations:** permette di configurare i seguenti parametri:
  - baud: configura la velocità della connessione seriale.
  - client: configura l'indirizzo IP della scheda ethernet di *Antares*.
  - server: configura l'indirizzo IP del server *TFTP*[1], utilizzato nei trasferimenti di file tramite ethernet.

[1]Trivial File Transfer Protocol

- gateway: configura l'indirizzo IP del gateway, qualora il server si trovi su una sottorete differente a quella dove è connessa la scheda *Antares*.
- netmask: maschera di rete. Definisce il pool di indirizzi che saranno considerati interni alla sottorete locale.
- filename: nome file di default che verrà utilizzato dal comando DN nei trasferimenti con protocollo *TFTP*.
- filetype: tipo di file che verrà utilizzato dal comando DN nei trasferimenti con protocollo *TFTP*. I possibili valori sono: "srecord", "coff", "elf", "image".

- **SHOW - Show Configurations:** elenca tutti i parametri di configurazione.

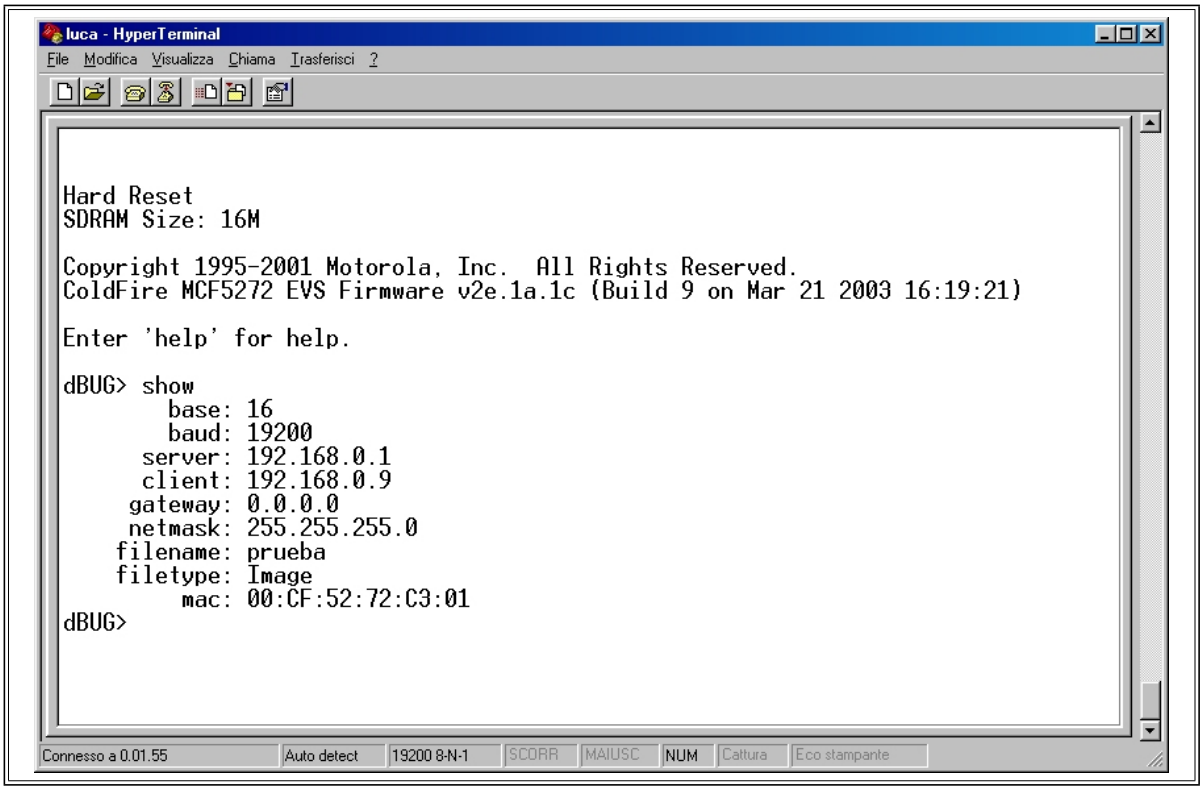


Figura 4.3: *Parametri di configurazione*

- **TRACE - Trace Into:** effettua l'esecuzione passo a passo del programma caricato in memoria.
- **UP - Upload Binary Data:** permette alla scheda di inviare un file tramite la connessione ethernet.

- **UPFLASH** permette la scrittura dati sul primo chip di memoria *flash*.
- **UPFLASH2** permette la scrittura dati sul secondo chip di memoria *flash*.
- **VERSION:** riporta la versione del programma dBUG.

Tutti questi comandi accettano parametri. La sintassi esatta di ciascun comando può essere consultata sul manuale [5] o puo essere dedotta dando il comando “HELP” alla scheda.

Capitolo 5

Installazione del sistema operativo μClinux

In questo capitolo verrà giustificata la scelta di usare *μClinux* come *sistema operativo* della piattaforma *Antares*. e ne verrà analizzata l'architettura *software*.

L'obbiettivo è quello di mettere il lettore in condizione di sviluppare applicazioni specifiche sulla piattaforma *Antares* e di modificare le opzioni del sistema operativo stesso, qualora lo ritenga necessario.

5.1 Sistema operativo

5.1.1 μClinux

Il sistema operativo *μClinux* è una versione del popolare sistema operativo *Linux* orientata ai microcontrollori che non dispongano di un'unità di gestione della memoria (*MMU*), come nel caso del *MCF5272*. Questo sistema operativo è caratterizzato dall'occupare uno spazio decisamente più modesto di quello di una convenzionale distribuzione *Linux*, fatto che lo rende idoneo allo sviluppo di sistemi embedded. A titolo di esempio, lo spazio occupato da un *kernel* *μClinux* configurato con le opzioni standard, può essere minore di *500 KBytes*, mentre quello di una distribuzione completa *μClinux*, fornita dei più comuni applicativi, può essere inferiore ai *900 KBytes*.

Come nel caso di molte altre distribuzioni di *Linux*, *μClinux* viene distribuito

sotto licenza *GPL*. Questo significa che l'intero sistema operativo è accompagnato dal relativo codice sorgente, modificabile e ridistribuibile nei termini della licenza *GPL* consultabile in [18]. Ciò è chiaramente uno dei maggiori vantaggi offerti da questo *sistema operativo*, in quanto ci permette di personalizzare completamente il sistema in modo che si adatti esattamente alle nostre necessità.

A quanto detto, bisogna aggiungere che il *kernel* di *Linux* offre al nostro sistema i notevoli vantaggi tipici di un qualsiasi computer dotato di *Linux*, ovvero:

- si tratta di un sistema operativo multitasking.
- è dotato di un'architettura modulare, che permette di configurare il kernel con le sole opzioni necessarie, riducendo lo spazio occupato.
- offre il supporto di molteplici protocolli di rete: *TCP*/*IP*, *PPP*, *SLIP*, etc.
- offre il supporto di molteplici fyle system: *NFS*, *Ext2*, *FAT32*, *FAT16*, etc.
- é robusto e affidabile.
- la disponibilità del codice sorgente permette di adattarsi alle necessità concrete di qualsiasi applicazione.
- é gratuito, dovuto al fatto che gran parte del software è distribuito sotto licenza *GPL*.
- permette la creazione di driver per supportare nuovo *hardware*. Questo punto è facilitato ulteriormente dall"esistenza di un gran numero di driver "open source", che possono essere pertanto adattati.

La *API*[1] del sistema operativo μ*Clinux* è la stessa di un sistema *Linux*, convenzionale, quindi adattare applicazioni *Linux* al sistema μ*Clinux* è un passo praticamente immediato. La distribuzione di μ*Clinux* inoltre è già predisposta per girare sui processori *Motorola MCF5272*, fatto che riduce ancora il numero di modifiche al codice sorgente che è stato necessario effetuare.

La programmazione per questo sistema operativo è realizzata tipicamente in linguaggio *C* o *C++*, anche se esistono implementazioni di macchine virtuali, che permettono lo sviluppo di applicazioni scritte in *Java*. Queste ultime risultano però più

[1] *Application Programming Interface*: In questo caso ci riferiamo all'insieme di chiamate al sistema operativo attraverso le quali si accede ai servizi offerti dallo stesso.

lente delle analoghe scritte in *C* e non verranno prese in considerazione in questa tesi.

Tuttavia, anche questo sistema operativo presenta i suoi inconvenienti. La sua principale carenza è identificabile nella gestione di applicazioni grafiche e di interfacce utente grafiche, dal momento che il loro sviluppo può risultare più complesso se comparato con altri sistemi operativi, come ad esempio *Windows CE*. Data la finalità del sistema *Antares* questo inconveniente è tollerabile, soprattutto se proporzionato ai molteplici vantaggi offerti.

Tutte le informazioni sulla distribuzione *μClinux* possono essere ottenuti sulla pagina internet ufficiale [10].

5.2 Architettura software del μClinux

In primo luogo, la distribuzione *μClinux* è composta da 2 grandi blocchi di software, drasticamente differenti.

Da un lato vi è il *kernel* del sistema operativo, che è dedicato alla gestione dell'*hardware* del sistema, alla gestione dei diversi processi, alla gestione dei files, etc.

Dall'altro vi sono i programmi di sistema, che offrono all'utente un'interfaccia amichevole verso i servizi forniti dal *kernel*. Tra questi vi sono, ad esempio, la "*shell*[2]" o le applicazioni per la gestione dei file, come *cp, mkdir, ...*.

Vi sono poi le applicazioni create dall'utente, che utilizzano sia i servizi del *kernel* sia i programmi del sistema e sono quelle che apporteranno una funzionalità concreta alla piattaforma in considerazione.

5.2.1 Schema di installazione.

Il sistema operativo *μClinux* e le applicazioni create dall'utente non sono però gli unici componenti software presenti sulla scheda *Antares*. Vi sono, infatti, due altri

[2]Si denomina *shell* l'"interprete dei comandi", utilizzato per dare ordini al sistema operativo. Questo nome deriva dalla denominazione stessa del programma, incaricato di "mascherare" il sistema operativo e presentare all'utente un'interfaccia più amichevole, seppur di natura testuale.

componenti necessari al corretto funzionamento della scheda: il firmware e il "*boot loader*"

Il firmware è una versione del "*dBUG*", un software *Motorola* che ho adattato all'*hardware* della nostra piattaforma e che viene ampiamente descritto nel capitolo 4. Il *dBUG* provvede a configurare l'*hardware* della piattaforma e offre un supporto minimale di comandi per gestire la scheda *Antares* in assenza del sistema operativo.

Il "*boot loader*" è il programma *"CoLiLo" (Coldfire Linux Loader)*, distribuito con licenza *GNU*, anch'esso adattato all'hardware di *Antares*. Il suo compito è quello di: recuperare dalla memoria *Flash* un'iMmagine compressa del sistema μClinux, caricarla nella memoria *RAM* ed avviare l'esecuzione del codice appena estratto.

Quando viene fornita alimentazione alla scheda *Antares* o dopo aver premuto il pulsante *Reset* posto sulla scheda stessa, il microcontrollore *MCF5272* provvede ad avviare il codice contenuto nel primo chip di memoria *flash*, vale a dire alla posizione 0xFFE00000[3]. In questa posizione di memoria risulta installato il firmware "*dBUG*", che provvederà ad inizializzare la scheda, configurare le linee di "*chip select*", la porta di comunicazione seriale e la porta ethernet.

Verrà quindi presentato sul terminale seriale connesso alla prima porta seriale un prompt dei comandi.

```
Hard Reset
SDRAM Size: 16M

Copyright 1995-2001 Motorola, Inc.  All Rights Reserved.
ColdFire MCF5272 EVS Firmware v2e.1a.1c (Build 9 on Mar 21 2003 16:19:21)

Enter 'help' for help.

dBUG> _
```

Sarà ora possibile utilizzare le molteplici funzioni offerte da questo firmware, descritto nel capitolo 4.

In particolare, con il comando "`colilo`" verrà richiamato il "*boot loader*", che dovrà essere installato nella seconda metà del primo chip di memoria *flash*, vale a dire in 0xFFF00000.

[3]Per la rappresenazione dei numeri interi in forma esadecimale verrà utilizzato il sistema tipico del linguaggio *C*,vale a dire, aggiungendo il prefisso"*0x*".

A seconda delle opzioni di compilazione del *CoLiLo*, ampliamente descritte nella sezione 5.4.1 potrà essere presentato un ulteriore prompt dei comandi, ma la semplice attesa di alcuni secondi porterà all'esecuzione del sistema operativo *μClinux*.

Il file contenente l'immagine compressa del sistema *μClinux* dovrà essere installato alla posizione 0xFFF10000, vale a dire nella parte finale del primo chip di memoria *flash*. La restante parte di memoria *flash* rimasta libera, vale a dire tra le posizioni 0xFFE30000 e 0xFFF00000, sarà utilizzata dal *μClinux* con un file system specifico per memorie *flash*, il "JFFS", descritto nella sezione 5.6.4.

Nella ***Tabella 5.1*** viene riassunta la suddivisione della memoria del primo chip di *Flash*, e viene anche riportata la nomenclatura data dal sistema operativo μClinuxai vari "block device".

Inizio	Fine	Dimensione	Contenuto	Device μClinux
0xFFE00000	0xFFE2FFFF	256 Kbyte	dBUG firmware	-
0xFFE30000	0xFFEFFFFF	832 Kbyte	JFFS fylesystem	/dev/mtdblock2
0xFFF00000	0xFFF0FFFF	64 Kbyte	CoLiLo	/dev/mtdblock0
0xFFF10000	0xFFFFFFFF	960 Kbyte	μClinux	/dev/mtdblock1

Tabella 5.1: *Mappa della prima flash*

Il secondo chip di *Flash* è completamente libero. L'utente potrà utilizzarlo per installare una seconda distribuzione *μClinux*.

5.3 Utility di sviluppo

In questa sezione vedremo le utility che sono state utilizzate per lo sviluppo di questo progetto e che risultano indispensabili a chi voglia sviluppare applicazioni per la piattaforma *Antares*. La postazione di lavoro usata non è altro che un personal computer, con una distribuzione del sistema operativo *Linux*, dove è stato installato il software necessario, ovvero il compilatore, l'assemblatore, il linker. Dal momento che l'architettura della scheda *Antares* è diversa da quella del PC della postazione

di lavoro, (tipicamente una *Intel x86*) sarà necessario utilizzare un "cross-compiler"[4] in modo da poter creare file binari adatti ad essere eseguiti su un'architettura differente da quella dove si lavora. Questo è il motivo per cui risulta impossibile usare i normali tool di compilazione forniti con una convenzionale distribuzione *Linux*.

5.3.1 Installazione

Descriverò ora il procedimento di installazione delle utility sopra citate. Innanzitutto è necessario un PC su cui sia già stata precedentemente installata e configurata una distribuzione del sistema operativo *Linux*. Nel mio caso è stata usata una distribuzione *Debian*.

Per facilitare la fase di installazione del "cross-compiler" si allega a questa tesi un *CD-ROM* nel quale vi sono i file binari precompilati per un'architettura generica *Intel x86*.

Il primo passo sarà quello di scompattare il file:
"`m68k-elf-tools-20020410.tar.gz`" in una directory provvisoria, per riottenere la struttura di file e directory appropriata. Per installare il compilatore "system wide", ovvero in maniera che sia utilizzabile da tutti gli utenti della macchina, sarà necessario possedere i diritti di amministratore del PC in questione, vale a dire conoscere la password dell'utente *root*. A questo punto dovremo copiare la struttura di cartelle precedentemente ottenuta nella diretory radice del nostro sistema. Eseguita questa operazione dovremo verificare che la variabile di sistema *PATH* contenga la directory "`/usr/local/bin`" in modo che il sistema possa raggiungere i file binari appena installati.

Di seguito riporto un elenco delle utility contenute nel pacchetto appena installato:

- *m68k-bdm-elf-gdb*: si tratta del cross-debugger, che ci permette di depurare i programmi, scritti per un'architettura *Coldfire*, usando una macchina *Intel x86*.

[4] Con il termine generico di "cross-compiler" si intende tutta una serie di strumenti necessari per la compilazione di codice, vale a dire il "cross-compiler", il "cross-assembler", il "cross-debugger", il "cross-linker"

- *m68k-elf-gasp; m68k-elf-as*: sono il preprocessore e l'assemblatore per processori della famiglia *Coldfire*.
- *m68k-elf-gcc; m68k-elf-c++; m68k-elf-g++*: si tratta del cross-compiler per i linguaggi *C* e *C++*.
- *m68k-elf-ld*: si tratta del linker, incaricato di generare i file eseguibili a partire dai file oggetto prodotti dal compilatore.
- *m68k-elf-ar; m68k-elf-ranlib*: queste utility permettono la creazione e la gestione di librerie a partire da vari file oggetto.
- *m68k-elf-objcopy; m68k-elf-readelf; m68k-elf-size; m68k-elf-objdump*: tramite l'uso di queste utility possiamo ottenere informazioni e manipolare i file oggetto.
- *m68k-elf-nm; m68k-elf-strip*: permettono di vedere e di eliminare la tabella dei simboli presente nei file oggetto.
- *m68k-elf-elf2flt*: trasforma un file di tipo *elf* in un file di tipo *flat*, che sono quelli che verranno utilizzati nel nostro sistema *μClinux*.
- *genromfs*: questa utility permette di creare il sistema di cartelle con gli applicativi che sono correlati al kernel di *μClinux*.

5.4 Il boot loader CoLiLo

In questa sezione analizzeremo in dettaglio il "boot loader". Come già accennato, tra le funzioni di questo software vi è quella di configurare l'*hardware* della scheda e provvedere a recuperare dal chip di *flash* l'immagine compressa della distribuzione *μClinux*. Per svolgere queste funzioni abbiamo utilizzato il *CoLiLo*, sviluppato inizialmente da *Rob Scott* e distribuito sotto licenza *GNU*.

Partendo da questa base sono state realizzate molteplici modifiche neessarie per adattare questo software all' *hardware* presente sulla scheda *Antares*. Inoltre sono state aggiunte numerose funzionalità come ad esempio la verifica del corretto funzionamento delle differenti parti dell'*hardware* stesso o la possibilità di avviare differenti immagini di *μClinux*.

5.4.1 Compilazione

In questa sezione illustrerò i passi basilari da seguire per compilare il programma *CoLiLo* e per copiarlo nella memoria *Flash* di *Antares*.

Nel *CD-ROM* allegato vi si trova il codice sorgente del *CoLiLo* nella sua versione già adattata all'*hardware* di *Antares*. Per effettuare correttamente la compilazione è necessario far uso del cross-compiler, che deve essere installato come descritto nella sezione 5.3.1.

Il processo di compilazione fa inoltre uso dell'utility *make*, disponibile in qualsiasi sistema *UNIX*, configurato per poter compilare codice. Il processo di compilazione, gestito da "*make*", realizza i seguenti passi:

- compila e linka il programma *CoLiLo*.
- genera una mappa del programma che viene salvata nel file "`colilo.map`".
- genera una versione binaria del programma: "`colilo.bin`".
- genera la versione *S-Record*[5] del programma: "`colilo.s19`".
- copia entrambe le versioni in una directory da dove risultano visibili al server *TFTP*[6].

La prima cosa da fare è modificare il file "`Makefile`" per indicare al programma *make* la directory condivisa tramite il server TFTP. Questo risulta utile in quanto uno dei sistemi per inviare il *CoLiLo* alla nostra scheda è di usare le funzionalità di client TFTP del firmware *dBUG*, che viene ampiamente illustrato nel capitolo 4. La variabile che dobbiamo modificare è "`TFTPDIR`" che deve puntare alla directory in questione. Ad esempio:

```
TFTPDIR = $(HOME)/tftpdir
```

Altre variabili da modificare sono quelle relative all'ubicazione delle utility per compilare viste nella sezione 5.3.1:

[5] il formato *S-Record* è un formato di file creato da *Motorola* per l'invio tramite porta seriale dei file ai suoi processori.

[6] Trivial File Transfer Protocol

```
CF_ROOT = /usr/local/lib/gcc-lib/m68k-elf/2.95.3
GCC_EXEC_PREFIX = /usr/local/bin/m68k-elf-
CC = $(GCC_EXEC_PREFIX)gcc
AS = $(GCC_EXEC_PREFIX)as
LD = $(GCC_EXEC_PREFIX)ld
OBJCOPY = $(GCC_EXEC_PREFIX)objcopy
```

Una volta eseguite queste modifiche sul file "Makefile" possiamo effettuare la compilazione del programma. Per fare ciò si dovrà semplicemente invocare il comando *make* dalla directory dove si trova il codice sorgente. Il processo di compilazione produce un'immagine del *CoLiLo* pronta per essere inviata alla memoria *Flash* di *Antares*. Occorre precisare che quest'immagine è stata linkata per essere salvata nella memoria *Flash* alla posizione *0xFFF00000*, che è ubicata alla metà della prima memoria *Flash* nella nostra scheda. Qualora risulti necessario compilare il *CoLiLo* affinchè lavori in un'altra posizione di memoria, occorrerebbe modificare il file rom.ld.

A questo punto è possibile decidere se compilare il *CoLiLo* in modalità interattiva o no, a seconda che si desideri che venga richiesto l'intervento dell'utente nella fase di avvio della scheda *Antares*.

Nel modo non interattivo, infatti, il *CoLiLo* recupera dalla memoria *Flash* l'immagine predefinita di *uClinux* e ne comincia l'esecuzione senza effettuare nessuna richiesta, mentre, nel modo interattivo, verrà presentata una linea di comando all'utente per permettergli di interagire con il programma.

La scelta tra queste due modalità di lavoro viene effettuata tramite la variabile "CONFIG_UI' passata come parametro al comando *make*. Questa variabile può assumere i valori *1* o *0* a seconda che si desideri o meno il comportamento interattivo.

Altra variabile di configurazione importante è quella che ci permette di scegliere su quale porta seriale della scheda *Antares* verrà stabilita la connessione. Questa è la variabile "CONSOLA", e può assumere i valori *1* o *2*, a seconda che si utilizzi la *UART0* o la *UART1*.

Vi è poi la possibilità di linkare il programma in due maniere differenti, in modo da poterlo eseguire dalla memoria *Flash* o dalla memoria *Ram*. Viene offerta questa possibilità per facilitare la fase di debug del programma *CoLiLo*, in quanto questa

risulta molto più agile se il software è stato linkato per funzionare in memoria *Ram*. La scelta tra *RAM* e *Flash* verrà effettuata tramite la variablie "FLASH", che dovrà assumere rispettivamente il valore *0* o *1*

A titolo di esempio, se desideriamo compilare un'immagine del *CoLiLo*, per salvarla nella memoria *Flash*, che comunichi sulla porta *UART0* e che richieda l'intervento dell'utente, basta invocare il processo di compilazione con la seguente chiamata:

```
# make FLASH=1 CONSOLA=1 CONFIG_UI=1
```

5.4.2 Istruzioni d'uso

Il *CoLiLo*, come anche il *uClinux*, utilizza per default la porta *UART0* del microcontrollore per comunicare con l'utente. Per poter comunicare con la scheda è necessaro un terminale seriale configurato nel seguente modo:

- velocità Tx/Rx: 19200 baud.
- 8 bit di dati senza parità (8N1).
- controllo di flusso: nessuno.

Come terminale useremo il nostro stesso PC, collegandolo ad *Antares* tramite un cavo seriale null-modem. Come software per instaurare la comunicazione con la scheda possiamo utilizzare *Hyperterminal* se stiamo usando un ambiente *Windows* o *Kermit* se invece lavoriamo in un ambiente *Linux*.

Una volta avviato il *CoLiLo*, otterremo una schermata simile alla seguente:

```
Antares v1.2 boot

1. Arrancar imagen de usuario (Por defecto)

2. Arrancar backup de uClinux

i. Entrar en modo interactivo

Arrancando en 3 segundos...
CoLiLo (Antares)> _
```

Ci viene presentata, in lingua spagnola[7], la possibilità di avviare una delle due immagini del sistema operativo salvate in memoria *Flash*, (tasti 1 e 2) oppure di ottenere un interprete dei comandi in modo interattivo (tasto i). La prima opzione sarà presa, di default, una volta trascorsi *5* secondi.

Se, una volta entrati nel modo interattivo, digitiamo il carattere '?', otterremo una breve tabella che illustra i differenti comandi e i relativi parametri che si possono dare al *CoLiLo*.

```
CoLiLo (Antares)>?

Coldfire Linux Loader
Comandos:
  b                          - arranca la imagen por defecto
  m [addr]                   - volcado de memoria desde [addr]
  g [xfer_addr]              - ejecutar desde [xfer_addr]
  l [down_addr] [port]       - cargar imagen a la direccion [down_addr]
mediante [port]
  c [source_addr] [dest_addr] [size] - copia una zona de memoria
  f [source_addr] [dest_addr] [size] - copia una zona de memoria en flash
  d [source_addr] [dest_addr] [size] - descomprime una imagen
  x [down_addr] [dest_addr] [xfer_addr] [port] - secuencia l, d, g
  z [source_addr] [dest_addr] [xfer_addr] - secuencia d, g
  t [source_addr] [final_addr] - comprueba memoria (RAM)
  s [value]                  - enciende los LEDs de proposito general
  a                          - realiza un Autotest de la placa

    Donde:
      xfer_addr: direccion en hexadecimal de comienzo de ejecucion
      down_addr: direccion en hexadecimal de descarga de la imagen
      dest_addr: direccion en hexadecimal de destino de la imagen
      source_addr: direccion en hexadecimal de la imagen origen
      final_addr: direccion en hexadecimal del final del test de memoria
      size: Tamano de la imagen
      port: 0 = /dev/ttyS0, 1 = /dev/ttyS1

  Valores por defecto:
      down_addr: fff10000
      source_addr: fff10000
      dest_addr: 20000
      xfer_addr: 20000
      size: 1581820
      port: 1
```

A seguire riporto una descrizione dei comandi appena citati.

[7] L'uso dello spagnolo è dovuto al semplice fatto che il software è stato sviluppato per l'"Universidad Politècnica de Madrid".

- **Comando "b":** inizia la sequenza di avvio con i valori di default, che sono specificati alla fine della schermata. Questi valori possono essere modificati all'atto di compilare il *CoLiLo*, mettendo mano al codice sorgente dello stesso.
  - *source_address:* indirizzo dove si trova l'immagine del sistema operativo che vogliamo caricare in memoria. Nel nostro caso *0xFFF10000*.
  - *dest_address:* indirizzo dove verrà copiata o decompressa l'immagine del sistema operativo. Nel nostro caso sarà la *0x20000*, dal momento che le posizioni di memoria antecedenti sono spazio di memoria riservato per il *CoLiLo* stesso.
  - *xfer_address:* indirizzo di memoria al quale si salterà non appena terminato di decomprimere l'immagine del sistema operativo, per avviarne l'esecuzione. Nel nostro caso sarà *0x20000*.
  - *size:* la dimensione del blocco di memoria da copiare. Non è necessario se il blocco di memoria è compresso, in quanto questa informazione si trova nello stesso file compresso.
- **Comando "m [addr]":** presenta il contenuto dei 64 byte seguenti la posizione data dal parametro *[addr]*.
- **Comando "g [xfer_addr]":** comincia a eseguire il codice situato nella posizione di memoria indicata dal parametro *[xfer_addr]*.
- **Comando "l [down_addr] [port]":** permette di inviare attraverso una delle due porte UART un'immagine alla memoria della scheda *Antares*. La porta utilizzata è indicata dal parametro *[port]*, mentre la posizione di memoria verso la quale mandare il file viene indicata dal parametro *[down_addr]*.
- **Comando "c [source_addr] [dest_addr] [size]":** copia da una zona di memoria ad un'altra.
- **Comando "f [source_addr] [dest_addr] [size]":** programma la memoria *Flash* che si trova alla posizione indicata con il contenuto della zona di memoria indicata dal parametro *[source_addr]*.
- **Comando "d [source_addr] [dest_addr] [size]":** questo comando decomprime un'immagine contenuta alla posizione di memoria indicata da *[sour-*

ce_addr] e la copia nella posizione *[dest_addr]*. L'algoritmo di compressione/decompressione utilizzato è quello di *Lempel-Ziv*. Vale a dire che per comprimere l'immagine utilizzeremo banalmente il comando *gzip* su un sistema *Linux*.

- **Comando "x [down_addr] [dest_addr] [xfer_addr] [port]":** realizza in modo sequenziale le chiamate ai comandi "l", "d" e "g".
- **Comando "z [source_addr] [dest_addr] [xfer_addr]":** realizza in modo sequenziale le chiamate ai comandi "d" e "g".
- **Comando "t [source_addr] [final_addr]":** effettua una verifica dello stato della memoria *RAM* compresa tra le posizioni passate come parametro
- **Comando "s [value]":** presenta sui *LED* generici della scheda *Antares* il valore dato come parametro.
- **Comando "a":** effettua una verifica semiautomatica di tutto l'*hardware* della scheda *Antares*. Detto test consiste nella realizzazione delle seguenti prove:
  - *verifica della memoria*.
  - *verifica dei* LED *generici*: vengono accesi in vari modi i *LED* situati sulla scheda.
  - *verifica delle porte digitali di ingresso/uscita:* per fare ciò è necessario connettere entrambe le porte tra loro con un cavo piano, per verificare che esista comunicazione tra le due, scrivendo nella *Porta 0* e leggendo nella *Porta 1*.
  - *verifica delle porte* UARTs*:* si devono connettere tra loro le *UART* tramite un cavo crociato e si verifica la comunicazione in entrambe le direzioni
  - *verifica del modulo modulatore (*PWM*):* è necessario collegare ai modulatori alcuni *LED* per verificare che si illuminino con varie intensità.
  - *verifica dellel linee di interrupt esterne:* è necessario connettere dei pulsanti alle linee di interrupt e verificare che il sistema riceva tale segnale

5.4.3 Descrizione tecnica

La configurazione della mappa di memoria della scheda *Antares* è illustrata nella ***Figura 5.1***.

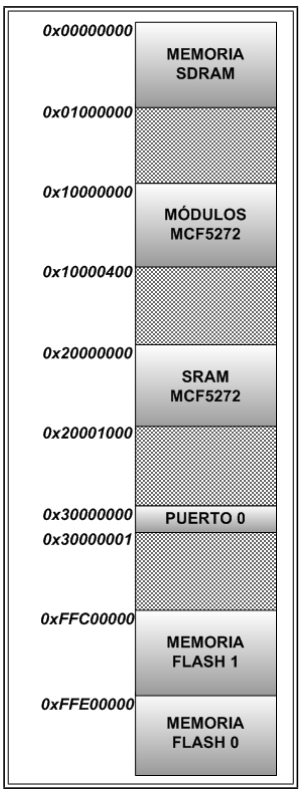


Figura 5.1: *Mappa di memoria Antares*

Come è già stato spiegato nella sezione 5.4.1, il *CoLiLo* recupera un'immagine dalla memoria *Flash0*, ovvero dalla poszione *0xFFE10000* e , dopo averla scompattata, la carica in *RAM*, nella posizione *0x20000*. Una volta caricata ne comincia l'esecuzione dalla medesima posizione *0x20000*.

5.5 Il sistema operativo μClinux

La distribuzione *μClinux* è nata per colmare la mancanza del sistema operativo *Linux* nel mercato dei sistemi embedded. Questa assenza era giustificata dal fatto che *Linux* è un sistema operativo avanzato che fa uso dei molteplici vantaggi offerti dalla unità di gestione della memoria (*MMU*), tipicamente è assente nella gran

maggioranza dei microcontrollori orientati a sistemi embedded. Questa particolarità ha reso molto difficile il procedimento di adattare il *kernel* di *Linux* a questo tipo di microcontrollori, in quanto, compiti molto semplici in presenza di microprocessori *MMU*, come ad esempio la protezione delle zone di memoria di ciascun processo, diventano invece compiti complessi con microcontrollori non dotati di *MMU*.

Il primo microcontrollore sul quale è stato eseguito un *μClinux* è il *Motorola MC68328* . Oggigiorno esistono "port" di questo sistema per un gran numero di microprocessori, tra i quali vi si trovano anche i microcontrollori della famiglia *Coldfire*.

Vediamo brevemente com'è organizzata la distribuzione *μClinux* e le modifiche che si sono rese necessarie sulla distribuzione originale per adattarla al nuovo *hardware*

5.5.1 Struttura della distribuzione

La distribuzione *μClinux* è formata da 3 blocchi di *software* principali:

- *il kernel:* in questa distribuzione sono inclusi i "port" del *kernel* di *Linux*. della versione *2.2.x* e della *2.4.x* . Verrà usata la versione più recente.
- *la biblioteca standard C (libc):* si trata di due implementazioni della biblioteca standard del C per microcontroller, la *uC-libc* e la *uClibc*. la differenza tra queste due, pur essendo minima, é dovuta al fatto che la *uC-libc* sebbene sia completa, non segue rigorosamente lo standard, per facilitare la migrazione di applicazioni specifiche per microcontrollori embedded. Di conseguenza è raccomandabile l'uso della biblioteca *uClibc*, più standard e completa, a meno che qualche applicazione non richieda specificamente la scelta della *uC-libc*
- *le applicazioni:* nella distribuzione *μClinux* sono incluse molteplici applicazioni, sia di sistema (come ad esempio l'interprete dei comandi), sia applicazioni per l'utente (server *Web*).

Una volta effettuato il processo di compilazione della distribuzione, viene generato un file binario unico contenente il *kernel* e il "file system" tipico di un qualsiasi sistema *Linux*, dove si possono trovare le applicazioni, le librerie ed i file di configurazione del sistema.

5.5.2 Modifiche alla distribuzione

In questa sezione analizzeremo le modifiche che è stato necessario effettuare per ottenere il funzionamento di *μClinux* sulla piattaforma *Antares*.

Come già specificato nell'appendice A, *Antares* è basata sulla scheda di sviluppo *Motorola M5272C3*; pertanto le modifiche sono state effettuate prendendo come punto di partenza il codice specifico per *M5272C3*. Le principali differenze tra queste due schede sono:

- **memoria Flash:** la scheda *Motorola* è provvista di un solo chip da *2 MBytes* di *Flash* del produttore *AMD* [1], mentre *Antares* è provvista di due chip da *2 MByte* ciascuno, del produttore *Fujitsu* [3], è stato pertanto necessario modificare il driver della *Flash*.
- **memoria SDRAM:**la scheda *Motorola* dispone di *8 MBytes*, mentre *Antares* è provvista di *16 Mbytes* de memoria *SDRAM*,
- **ingressi/uscite:** la scheda *Antares* è dotata di un insieme di porte di ingresso e uscita dati, come ad esempio le due porte da 8 bit digitali, o i *LED* generici. In questo caso è stato necessario scrivere driver specifici.

Con la distribuzione così ottenuta il processo di compilazione risulta semplicissimo, in quanto è sufficiente selezionare il supporto per la nostra scheda in fase di configurazione del *kernel*, come si può vedere nella figura ***Figura 5.2***.

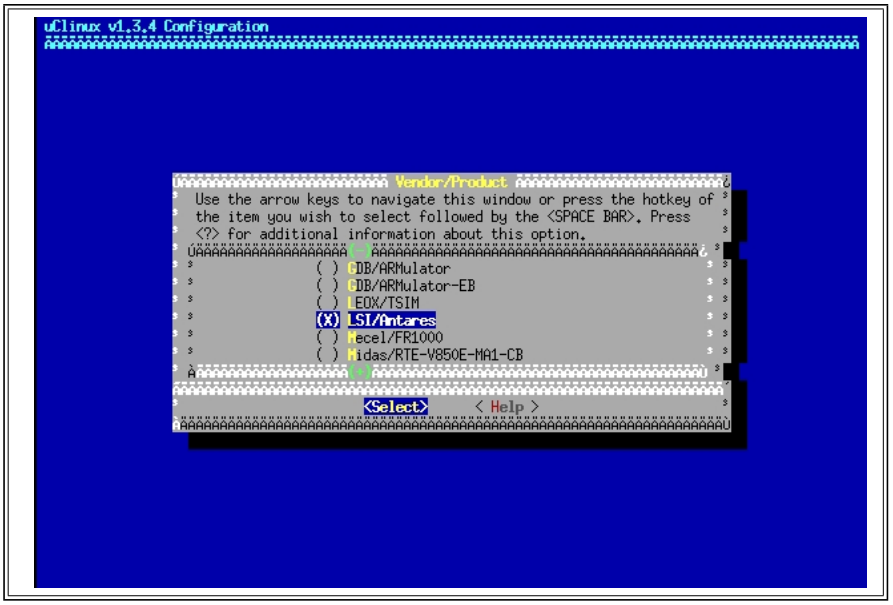


Figura 5.2: *Schermata di configurazione del μClinux*

5.5.3 Compilazione della distribuzione

Indichiamo, in maniera sintetica, i passi per effettuare la compilazione della distribuzione di *uclinux* a partire dal codice sorgente incluso nel *CD-ROM* allegato.

Il primo passo è quello di decomprimere il file contenente il codice della distribuzione. Otterremo una directory chiamata "*uClinux-dist*/", dove vi è la distribuzione completa. Nei paragrafi seguenti, parlando del processo di compilazione si farà sempre riferimento a sottodirectory di "*uClinux-dist*/"

A questo punto sarà necessario eseguire i seguenti passi:

1. configurazione tipo di scheda.
2. configurazione parametri del kernel.
3. Selezione e configurazione delle aplicazioni.
4. compilazione del *kernel*.
5. generazione di un'immagine binaria del kernel.
6. compilazione delle applicazioni.
7. creazione di un "file system" contenente le applicazioni.

8. generazione di un'immagine binaria che include il *kernel* e le applicazioni.

9. copia dell'immagine binaria della distribuzione completa ad una directory condivisa tramite un server *TFTP*, in modo da poterla inviare alla scheda *Antares*.

Per poter compilare il sistema dovremo aver installato sul nostro pc le utility viste nella sezione 5.3.1, cosi come l'utility "`make`".

Per avviare il processo di configurazione e compilazione dovremo usare una delle seguenti stringhe di comando, a seconda del sistema che desideriamo usare: testuale, testuale con ausilio di menù o grafico (necessita di un ambiente grafico X).

```
# make config
# make menuconfig
# make xconfig
```

Nel nostro caso abbiamo utilizzato la seconda opzione, che permette di usare un ambiente amichevole (grazie all'utilizzo delle librerie ncurses), senza dover scomodare un'interfaccia grafica.

Le scelte più importanti per la configurazione del sistema sono: la scelta della piattaforma per la quale stiamo compilando, ovvero *Antares*, sviluppata dall'*LSI*, la versione del *kernel* , ovvero la *2.4.x* e la versione della libreria *C*, nel nostro caso la *uClibc*. Sarà poi necessario selezionare

- Customize Kernel Setting
- Customize Vendor/User Settings

per poter in seguito accedere ai relativi menu di configurazione.

Tutte queste opzioni sono illustrate nel menù di configurazione della scheda, riportato in figura ***Figura 5.3***.

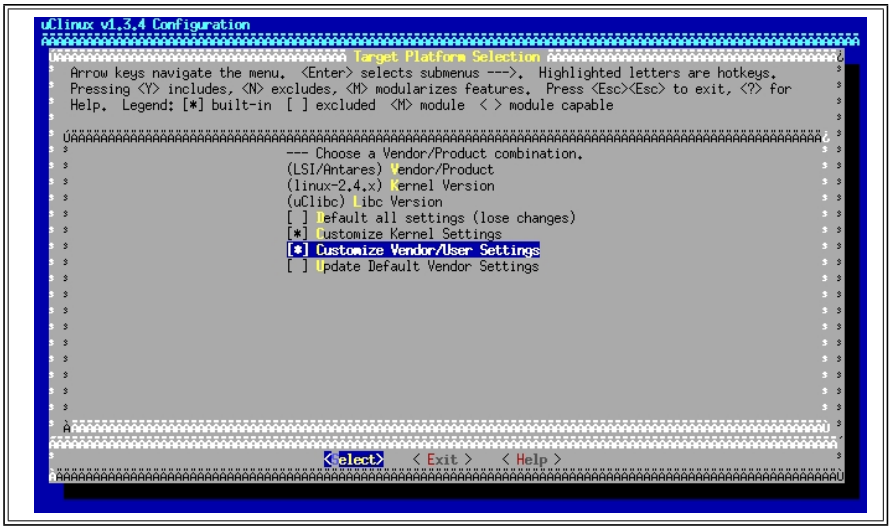


Figura 5.3: *Schermata di configurazione del* uclinux.

A questo punto, procedendo, possiamo personalizzare la configurazione del *kernel* e delle applicazioni.

Tra le varie cose che si possono configurare, troviamo la scelta del modello di microprocessore, ovvero il *MCF5272*, il tipo di interfaccia *Ethernet*, il driver del modulo *QSPI*, i driver della memoria *Flash*, il supporto per vari filesystem, tra cui *NFS*, *JFFS*, etc.

Nella ***Figura 5.4*** possiamo vedere la schermata principale della configurazione del *kernel* di *μClinux*.

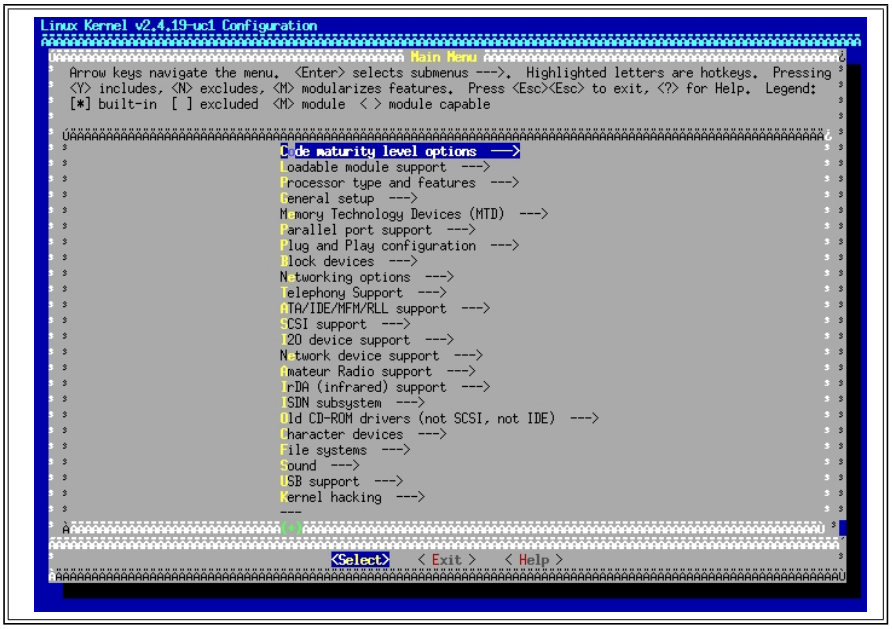


Figura 5.4: *Schermata di configurazione del kernel di μClinux*

Una volta terminata la configurazione del *kernel*, passeremo a selezionare e configurare le applicazioni che saranno incluse nella distribuzione. Tra queste, a titolo di esempio, troviamo l'interprete dei comandi e programmi di sistema, come "`cp`" o "`mount`".

Tra i vari tool vi troveremo inoltre: un insieme di utility che ci permetteranno di maneggiare la memoria *Flash* ed il filesystem che vi sarà implementato, un insieme di programmi mirati a sfruttare le capacità di comunicazione della scheda (programmi per gestire la posta elettronica, server *Web* e *FTP*).

Tutte queste applicazioni renderanno molto semplice creare sistemi embedded che facciano uso in maniera assolutamente convenzionale delle reti di comunicazione esistenti e che abbiano accesso ad *Internet* e a tutti i vantaggi da quest'ultimo offerti.

Gli ultimi passi consistono nel determinare le dipendenze tra i vari tool e compilare. Per fare ciò eseguiremo i seguenti comandi:

```
# make dep
# make
```

Terminato ciò, se tutto sarà stato compilato senza errori, sarà stato generato un file con il *kernel* e le applicazioni che precedentemente abbiamo selezionato. Que-

st'immagine si trova nella directory "`images/`" e nella directory condivisa con il server *TFTP*. Il nome del file é "`image.bin`".

A questo punto occorrerà inviare l'immagine alla memoria *Flash*. Per fare ciò è necessario comprimere il file "`image.bin`" con l'utility *gzip* e copiarla nella posizione *0xFFC10000* o nella posizione *0xFFF10000*, per poter successivamente scompattarla e avviarla grazie al *CoLiLo*.

5.6 Filesystem di μClinux

Il sistema operativo μClinuxsupporta svariati tipi di "filesystem", tra cui tutti i tipi più comuni. In un sistema "embedded", quale la scheda *Antares*, la scelta dei "filesystem" determina drasticamente le prestazioni del sistema. È determinante, infatti, per un sistema di questo tipo, occupare la minima quantità di memoria possibile, e permettere la scrittura di dati nelle memorie di tipo non volatile.

5.6.1 "filesystem" RAMFS

Si tratta di un filesystem che utilizza come dispositivo di salvataggio dati una parte della memoria *RAM*. Per utilizzare questo "filesystem" è necessario abilitare il supporto per il "*RAMDISK*" in fase di compilazione del *kernel*.

Su questo tipo di filesystem verranno memorizzati i dati contenuti nella directory "`/tmp`".

5.6.2 "filesystem" NFS

Con la sigla "*NFS*" si indica il (*Network File System*). Si tratta di un "filesystem" remoto che ci permette di "montare" una cartella di un computer remoto, accessibile tramite la scheda *ethernet*. La directory cosi montata verrà usata esattamente come se fosse una directory locale.

Il sistema "*NFS*" dovrà essere configurato, come descritto nella sezione 5.7.2

5.6.3 "filesystem" ROMFS

Si tratta di un "filesystem" di sola lettura, molto semplice, specifico per dispositivi di memoria non volatile, adattissimo all'uso su sistemi "embedded" per il bassissimo spreco di spazio che comporta.

5.6.4 "filesystem" JFFS

Si tratta del "*JFFS*" (*Journaling Flash File System*), pensato per lavorare su dispositivi di memoria non volatile come la memoria *Flash*. Si tratta di un "filesystem" di lettura e scrittura, che permette la scrittura di file in maniera assolutamente trasparente, nonostante sia parecchio laboriosa la gestione della scrittura su di una memoria *Flash*.

5.7 Lavorando con μClinux

Vediamo ora i passi fondamentali per configurare il sistema e cominciare a lavorare sulla piattaforma. Saranno descritte la configurazione dei parametri di rete e l'aggiornamento della distribuzione *μClinux*, che è salvata sulla memoria *Flash*.

5.7.1 Configurazione dell'interfaccia Ethernet

La prima cosa da fare all'atto di avviare il nostro sistema è assegnare un indirizzo *IP* all'interno della nostra rete locale. Sarà inoltre importante configurare l'indirizzo del *gateway* della nostra rete, per permettere alla nostra scheda di accedere a reti remote.

L'interfaccia di rete si configura alla stessa maniera di un sistema *Linux*, vale a dire facendo uso del comando "*ifconfig*".

Supponiamo di dover configurare la nostra scheda nel seguente modo:

- *indirizzo IP:* 138.4.9.100
- *netmask:* 255.255.255.192
- *indirizzo di broadcast:* 138.4.9.127

- *indirizzo del gateway:* 138.4.9.65

Sarà necessario configurare la scheda con i seguenti comandi:

```
# ifconfig eth0 down
# ifconfig eth0 138.4.9.82 broadcast 138.4.9.127 netmask
  255.255.255.192
# ifconfig eth0 up
# route add default gw 138.4.9.65
```

Sarà ora possibile l'accesso a tutti i tipici servizi *IP*, come, ad esempio, la posta elettronica, i servizi *Web*, *telnet*, etc.

Di particolare interesse sarà il servizio *telnet*, che ci permetterà di connetterci alla scheda *Antares* remotamente, senza dover usare la connessione seriale.

5.7.2 Accesso a directory remote tramite NFS

Per poter accedere ad una directory tramite *NFS* dovremo configurare sia il client *NFS*, sia il server.

- *Configurazione del server:* nel computer che dovrà esportare la directory tramite *NFS* dovrà essere installato un server *NFS*[8]. Il sistema si configura tramite il file `/etc/exports` al quale sarà necessario aggiungere una linea simile alla seguente:

  ```
  /home/necchi/mydir    138.4.9.98/26(rw)
  ```

  Nell'esempio abbiamo configurato il server *NFS* in modo che accetti connessioni dall'intera sottorete 138.4.9.64 - 138.4.9.127, con permessi di lettura e scrittura sulla directory : "*/home/necchi/mydir*".

- *Configurazione del client:* come per qualsiasi altro filesystem, l'operazione di montaggio verrà effettuata tramite il comando "*mount*", incluso nella distribuzione.

  Sarà necessario introdurre al prompt dei comandi di *antares* la seguente linea di comando:

[8] In *Linux*, il server del servizo *NFS* è il programma "*nfsd*".

```
# mount -t nfs -o nolock 138.4.9.98:/home/necchi/mydir /mnt
```

Una volta montata la directory in questione, sarà possibile utilizzarla esattamente come se si trattasse di una directory locale.

5.7.3 Uso di un filesystem JFFS

Vedremo ora come sia possibile utilizzare un filesystem *JFFS* sulla memoria *Flash*.

I driver *MTD* della memoria *Flash* definiscono delle partizioni sul chip di memoria, che vengono mappate tra i device a blocchi del sistema operativo.

Nella tavola di partizioni presentata nella tabella ***Tabella 5.1*** vediamo che la *partizione 2* è quella destinata al filesystem *JFFS* e che pertanto dovremo utilizzare il dispositivo a blocchi "*/dev/mtdblock2*".

- *Creazione del filesystem:* Per fare ciò dovremo utilizzare alcuni programmi raccolti nel pacchetto "*mtd-utils*", che devono essere stati inclusi nella configurazione del kernel in fase di compilazione e che permettono di realizzare molteplici operazioni sulla memoria *Flash* e sulle sue relative partizioni. Per cancellare il contenuto della partizione useremo il comando "*eraseall*", che richiede come parametro il nome del dispositivo a blocchi relativo. Per creare il filesystem faremo analogamente uso del tool "*mkfs.jffs*".
- *Montaggio del filesystem JFFS:* sarà sufficiente utilizzare il seguente comando:

```
# mount -t jffs -o nolock /dev/mtdblock2 /flash
```

5.7.4 Aggiornamento del μClinux

Grazie ai driver *MTD* della *Flash* e alla struttura di partizionamento di tale dispositivo, sarà possibile sostituire l'immagine compressa della distribuzione *μClinux* presente sulla scheda *Antares*. Per fare ciò ricorreremo al tool "*dd*"

Una volta ricompilata la distribuzione di μClinux, come indicato nella sezione 5.5.3, basterà copiare il file binario in una directory condivisa tramite *NFS* e digitare un comando simile al seguente:

```
# dd if=/mnt/image.bin.gz of=/dev/mtdblock1
```

Dove /mnt/image.bin.gz è il file contenente l'immagine del sistema compressa tramite *gzip*.

Al prossimo riavvio del sistema verrà avviata la nuova versione della distribuzione *μClinux*. Il procedimento appena descritto può essere anche usato per aggiornare il programma *CoLiLo*.

Capitolo 6

Conclusioni

Le varie componenti software e hardware, descritte in questa tesi, sono state testate verificandone l'effettivo funzionamento e la corrispondenza con le specifiche di progetto.

Per poter avviare le esercitazioni di laboratorio su questo nuovo ambiente didattico si dovrà attendere la realizzazione, da parte dell'"*LSI*" (Laboratorio di Sistemi Integrati), di un numero adeguato di schede *Antares*, *hardware* che si sta dimostrando, per le sue prestazioni e la sua versatilità, un componente essenziale per la didattica e la ricerca presso il dipartimento di Ingegneria Elettronica dell'Università di *Madrid*.

Una linea di sviluppo futura potrebbe richiedere l'aggiunta del supporto multilingue, al fine di poterlo proporre anche agli studenti del *Politecnco di Torino*.

Appendice A

Descrizione della piattaforma Antares

In questo capitolo verrà descritta la piattaforma *Antares*. Questa scheda, visibile nella ***Figura A.1***, è stata recentemente creata dai ricercatori e dai tesisti del dipartimento *DIE*, "Dipartimento di Ingegneria Elettronica" presso il "laboratorio di Sistemi Integrati", denominato "*LSI*", dell'"Universidad Politecnica de Madrid".

Antares è stata progettata tenendo come riferimento la scheda di sviluppo *M5272C3* di *Motorola*, pertanto è basata su un processore della famiglia *Coldfire*. Le caratteristiche generali di *Antares* sono le seguenti:

1. microcontrollore 32 bits *MCF5272 Motorola*.
2. memoria *SDRAM* (16 Mbytes).
3. memoria *Flash* (4 Mbytes).
4. interfaccia *Ethernet* per connessioni ad una rete locale.
5. due interfacce seriali *RS-232*.
6. una interfaccia *USB* slave.
7. una interfaccia *BDM*/*JTAG* per effettuare debug e per programmare dall'esterno le due unità di memoria *Flash*.

8. otto uscite digitali *LCMOS*[1].
9. otto entrate/uscite *LCMOS*.
10. quattro *LED* indicadori.
11. bus de espansione.
12. alimentazione di entrata di *8-14 Volt*.

Nella ***Figura A.1*** possiamo vedere la piattaforma di sviluppo *Antares* nella quale sono state evidenziate e numerate le differenti parti che la compongono.

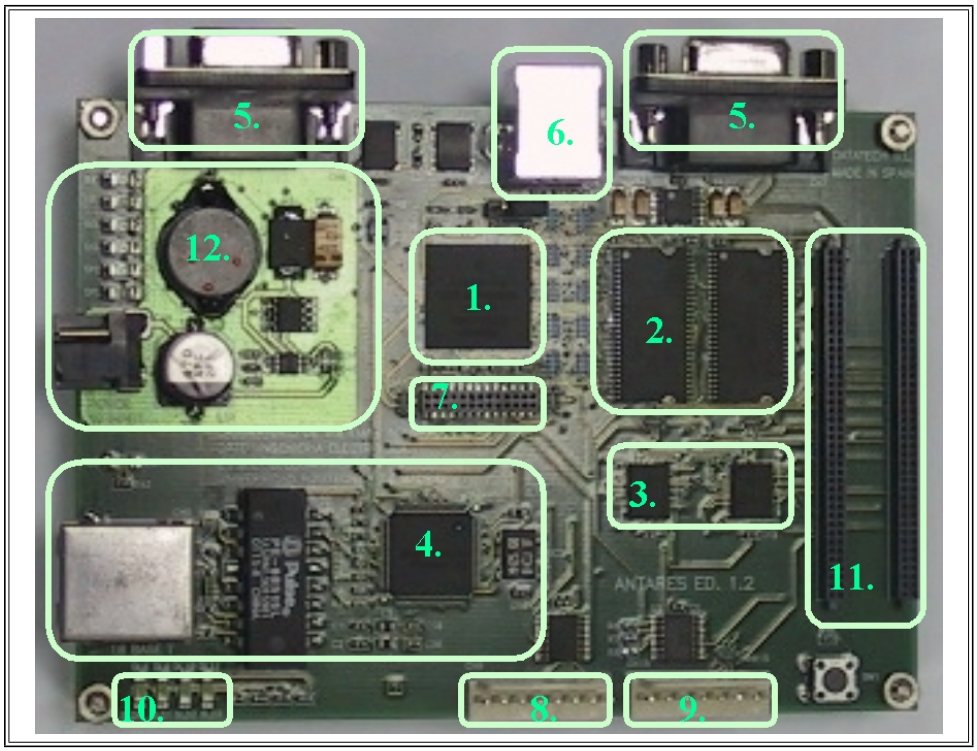


Figura A.1: *Piattaforma di sviluppo Antares*

Antares è caratterizzata da una notevole densità di integrazione, fattore che ne permette dimensioni notevolmente ridotte. La scheda infatti misura soli $12cm \times 9cm$.

Segue la descrizione dettagliata dei moduli che compongono la piattaforma *Antares*.

[1]La famiglia logica LCMOS è pensata per ottenere bassi livelli di consumo, utilizzando una tensione di alimentazione di *3.3 V*.

A.1 Microprocessore

Il Microprocessore è una delle parti più importanti della scheda, essendo il componente che maggiormente influenza le prestazioni dell'intero sistema. Per la piattaforma *Antares* è stato scelto un microcontrollore della famiglia *Coldfire* di *Motorola*. Si tratta di microcontrollori[2] a 32 bits, *RISC*[3], caratterizzati dal poter lavorare con istruzioni di lunghezza variabile. Questo significa che le istruzioni accettate da questo microcontrollore possono avere lunghezza di 16, 32 o 48 bit. Ciò si traduce, se paragonato con quello delle architetture *RISC* a 32 o 64 bits, in un uso più efficiente della memoria, riducendo la larghezza di banda necessaria nei cicli di lettura delle istruzioni e aumentando le prestazioni globali del sistema.

La famiglia *Coldfire*, orientata al mercato dei sistemi embedded, è di fatto la diretta evoluzione della famiglia *68000*. Ciò rappresenta un notevole vantaggio, in quanto permette una migrazione semplice dei precedenti sistemi embedded implementati su di un'architettura *68000*. A tal proposito merita notare che il primo processore di questa nuova famiglia, che fu nominato "Il microcontrollore ponte verso la famiglia *Coldfire*", permetteva l'esecuzione sia di codice specifico per 68000, sia di codice specifico per *Coldfire*.

La famiglia *Coldfire* si compone di vari modelli. Contrariamente con quanto accade con i processori dei personal computer, (tipicamente *Intel* o *AMD*) nei quali la differenza tra ogni processore si basa sostanzialmente sulla frequenza di clock dello stesso, *Motorola* differenzia i vari prodotti in funzione del mercato a cui sono destinati. Questa politica trova la sua spiegazione nel tipo di mercato verso il quale si dirige *Motorola*. Si tratta, infatti, di un'offerta orientata a coprire le necessità di mercati molto piccoli e concreti, piuttosto che offrire una soluzione generica ad un mercato globale.

[2]Motorola definisce i processori della famiglia *Coldfire* come *processori embedded* e non come microcontrollori. Sebbene possano essere trovate argomentazioni per la differenziazione, questa denominazione obbedisce più ad una strategia di marketing che ad una differenza reale. In questo testo si utilizzeranno indifferentemente entrambe le demoninazioni.

[3]Le architetture *RISC* (Reduced Instruction Set Computer) si basano sull'implementazione di un numero ridotto di istruzioni semplici, che vengono seguite nella larga maggioranza in un solo ciclo do clock. Per contro, le architetture *CISC* (Complex Instruction Set Computer) offrono un maggior numero di istruzioni, che richedono in generale più di un ciclo di clock per la loro esecuzione.

A titolo di esempio, il modello *MCF5272* è orientato ad applicazioni di comunicazione, come router, gateway, mentre il modello *MCF5249* è diretto ad applicazioni che richiedano trattamento di segnali e richiedano l'accesso a dispositivi di salvataggio dati, come ad esempio un riproduttore *MP3*. È importante notare che, a prescindere da finalità tanto differenti, entrambe i modelli possiedono la medesima versione del nucleo *Coldfire* e sono quindi architetturalmente molto simili. Nella ***Figura A.2*** sono elencati i differenti modelli di *Coldfire*, con le relative caratteristiche tecniche peculiari.

Product	Core	Dhrys 2.1 MIPS @ max MHz	Processor Cache (bytes)	Processor SRAM (bytes)	Serial Interface, UART	Timers/ CS/ GPIO	DMA	DRAM Controller	10/100 Eth/ USB1.1	Operating Voltage (V)	Operating Frequency (MHz)	Temp (-40° C to +85° C)	Packaging	Rev	Additional Information
MCF5206	V2	17	512 I	512	2 UARTs	2/8/8	n/a	FPM, EDO	n/a	5	16, 25, 33	C	160-pin QFP	A	n/a
XCF5206e	V2	50	4K I	8K	2 UARTs	2/8/8	2-CH	FPM, EDO	n/a	3.3	40, 54	C	160-pin QFP	n/a	Enhanced pin-compatible version of 5206 with MAC, HW divide, BDM, I^2C, 5V tolerant I/O.
MCF5249	V2	125	8K I	96K	2 UART, I^2C, QSPI	2/4/up to 47	4-CH	SDRAM	n/a	1.8, 3.3	140	C	160-ball MAPBGA	n/a	EMAC, HW divide, BDM, 12-bit ADC, CDROM block. CD text, Hard Disk Drive, Memory stick interfaces. Audio decoders.
MCF5249L	V2	107	8K I	96K	2 UART, I^2C, QSPI	2/3/up to 34	4-CH	SDRAM	n/a	1.8, 3.3	120	C	144-pin LQFP	n/a	EMAC, HW divide, BDM, 12-bit ADC, CDROM block. Hard Disk Drive Interface. Audio decoders.
MCF5272	V2	63	1K I	4K	10/100 FEC, 2 UARTs, USB, QSPI	4/8/up to 32	2-CH	SDRAM	MAC/ MAC+PHY	3.3	66	C	196-ball MAPBGA	n/a	MAC, HW divide, BDM, 4 TDM GCI/IDL ports, software HDCL module, QSPI, 3 PWMs, 5V tolerant I/O.
PCF5280	V2	59	2K I	64K	3 UARTs I^2C, QSPI, FlexCAN	4 Timers, +4 DMA Timers, 7 Chip Sel., Up to 150 I/O's	n/a	SDRAM	MAC (FEC) / n/a	3.3, 5	66	C	256-ball MAPBGA	n/a	Enhanced CAN 2.0B controller. Flashless version of PCF5282.
PCF5282	V2	59	2K I	64K	3 UARTs I^2C, QSPI, FlexCAN	4 Timers, +4 DMA Timers, 7 Chip Sel., Up to 150 I/O's	n/a	SDRAM	MAC (FEC) / n/a	3.3, 5	66	C	256-ball MAPBGA	n/a	Enhanced CAN 2.0B controller, 512KB Flash. This product incorporates SuperFlash® technology licensed From SST.
MCF5307	V3	75	8K U	4K	2 UARTs, I^2C	2/8/16	4-CH	SDRAM, FPM, EDO	n/a	3.3	66, 90	C	208-pin FQFP	B	MAC, HW divide, BDM, PLL, I^2C, 5V tolerant I/O.
MCF5407	V4	316	16K I, 8K D	4K	UART, USART, I^2C	2/8/16	4-CH	SDRAM, FPM, EDO	n/a	1.8, 3.3	162, 220	C	208-pin FQFP	A	Pin-compatible 5307 performance upgrade with MAC, HW divide, BDM, PLL, I^2C, 3.3V tolerant I/O.

Figura A.2: *Modelli della famiglia Coldfire della Motorola*

Quando questa scheda è stata progettata, si potevano trovare in commercio tutti i modelli elencati ad eccezione delle due versioni del processore *5282*, che erano ancora in fase di produzione e che sono disponibili sul mercato solo da pochi mesi. Di tutti i modelli disponibili, quello che maggiormente si adatta alle necessità e ai requisiti di progetto di questa scheda è il *MCF5272*, in quanto integra in un solo chip tutti i sistemi di comunicazione verso l'esterno di cui possiamo necessitare. Come già detto, infatti, questo modello è orientato verso applicazioni dove si richiedano notevoli capacità di comunicazione. Anche il Microcontrollore *MCF5282* risponde ai

requisiti ed è verosimile che possa venir usato in una futura versione della nostra piattaforma *Antares*.

Di seguito illustro le caratteristiche generali del microcontrollore *MCF5272*.

A.1.1 Caratteristiche del *MCF5272*

Il *MCF5272* implementa, in un solo chip, tutti i protocolli di comunicazione che possiamo necessitare, offrendo molteplici interfacce, quali la *Ethernet*, o *SPI*. Questo processore è infatti indicato per tutte le applicazioni orientate alla comunicazione e alla trasmissione dati su *internet*, *telefonia*, reti *LAN*, etc. Come si può notare nella ***Figura A.3***, il microcontrollore *MCF5272* è formato da molteplici blocchi funzionali:

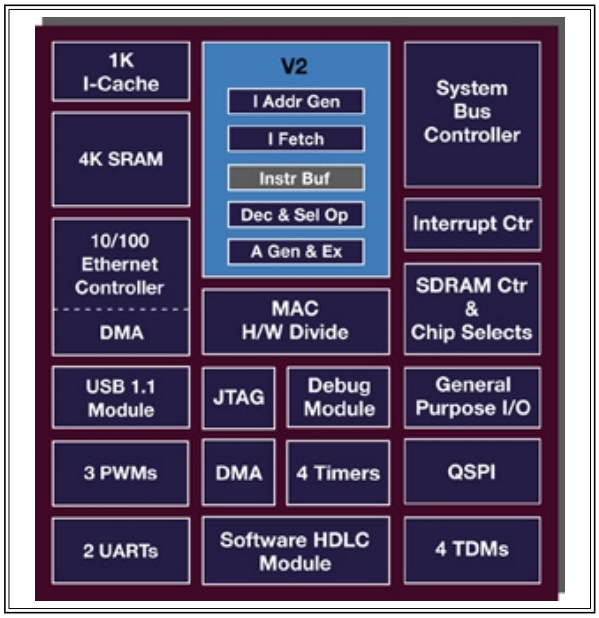


Figura A.3: *Diagramma a blocchi del MCF5272*

- *processore Coldfire V2:* il microcontrollore ha come processore un nucleo *Coldfire* versione 2. Si tratta, come già detto, di un processore *RISC* da 32 bits con una velocità di clock di 66 MHz e con una unità di *moltiplicazione/somma* (MAC), oltre ad un modulo per la relizzazione di divisioni tra interi. Permette inoltre l'esecuzione di istruzioni di lunghezza variabile (16, 32 o 48 bit). Questa è la prima versione di un nucleo *Coldfire* propriamente detto, in quanto il nucleo precedente, quello del *MCF5102* era un ibrido di architettura *Coldfire* e *68000*.

- *1KByte di cache:* questa memoria permette di aumentare il rendimento del controller in quanto si tratta di una memoria molto rapida e vicina al nucleo di calcolo. Ciò, ovviamente, permette di aumentare la velocità di esecuzione di strutture di codice ripetitive (come i cicli), in quanto immagazzina le istruzioni, che altrimenti dovrebbero essere sempre richiamate dalla memoria centrale.

- *4KBytes di memoria interna SRAM:* si tratta di memoria generica, che può essere usata nello stesso modo di quella esterna, con la sola ovvia differenza che, essendo molto più rapida, risulta utile per eseguire le parti di codice più critiche per quanto riguarda il tempo di esecuzione (calcoli in tempo reale, elaborazione audio, etc...).

- *modulo di debug:* il controller offre un modulo che permette di effettuare il debug del codice, attraverso l'interfaccia *JTAG*/*BDM*, fatto che facilita notevolmente il lavoro di sviluppo software per questo tipo di microcontrollore.

- *controller ethernet 10/100:* il microcontrollore include un controller *ethernet* 10/100 che permette la conessione del sistema a reti locali.

- *Controller USB 1.1 slave:* Controller USB 1.1: Questo blocco implementa lo standard *USB 1.1* e permette al controller di connettersi con altri dispositivi, sempre che ve ne sia uno che agisca da master (host).

- *modulo QSPI:* questo modulo permette di controllare fino a 16 dispositivi con interfaccia *SPI*, potendo agire da master, mentre non può agire da slave (il modulo non può essere controllato dall'esterno attraverso *SPI*)

- *controller SDRAM:* grazie al controller *SDRAM* integrato, possiamo dotare il nostro sistema di memoria *SDRAM*, che risulta molto più economica della più classica *SRAM*.

- *tre uscite PWM:* il controller possiede tre modulatori per ampiezza di impulsi (*PWM*) configurabili in maniera indipendente.

- *quattro porte TDM:* le quattro porte di multiplex per divisione nel tempo (*TDM*) permettono di collegare questo microcontrollore ad interfacce *RDSI*.

- *due UARTs:* il *MCF5272* offre due interfacce asincrone (*UART*), che permettono la comunicazione con altri dispositivi, come un terminale o un modem.
- *un canale DMA:* il canale di accesso diretto alla memoria (*DMA*) permette di pianificare trasferimenti di dati tra distinte zone di memoria o dispositivi senza richiedere l'intervento diretto del processore.
- *otto "chip-selects":* questo modulo permette di connettere al medesimo bus fino ad otto dispositivi o memorie senza richiedere l'aggiunta di nessuna logica addizionale.
- *tre entrate/uscite generiche da 8 bit:* rende disponibili un massimo di 48 bit di entrata/uscita, otto delle quali sono fisse, mentre le rimanenti sono multiplexate con altre funzioni del microcontroller.
- *quattro temporizzatori da 16 bit:* grazie a questo modulo si possono realizzare processi di controllo periodico in modo molto semplice.
- *watchdog software:* questo modulo é molto importante per i sistemi embedded, perchè permette di capire se il sistema non funziona correttamente e di riavviarlo nel caso ve ne sia la necessità. Questo è di particolare utilità quando si ha a che fare con sistemi di controllo, in quanto permette di arrestare con sicurezza il sistema nel caso venga rilevato un mal funzionamento.

Come si può notare il *MCF5272* ha molte caratteristiche peculiari particolarmente utili per la realizzazione di sistemi embedded che richiedano notevoli prestazioni nell'ambito delle comunicazioni. Per informazioni più dettagliate sui moduli qui descritti e sul loro funzionamento, si consulti il manuale [7].

A.2 Memoria

La scelta della memoria della piattaforma è quasi altrettanto importante alla scelta del processore stesso, perchè, analogamente a questo, influisce notevolmente sul rendimento del sistema. Le potenzialità del microcontrollore possono essere drasticamente limitate dalla memoria

utilizzata perchè il processore deve realizzare continuamente accessi alla memoria per ottenere le seguenti operazioni da eseguire. Sotto questo punto di vista il parametro che gioca il ruolo più importante é il tempo di accesso alla memoria.

A seguire saranno descritte le caratteristiche delle memoria utilizzate sulla nostra piattaforma (tanto quelle della memoria volatile, quanto quelle della memoria non volatile).

A.2.1 Memoria SDRAM

Approfittando della presenza del controller *SDRAM* integrato nel *MCF5272*, abbiamo dotato il sistema di questo tipo di memoria. Il controller integrato utilizza la linea di *chip-select 7* del microcontrollore per il controllo di detto banco di memoria *SDRAM*. *Antares* è dotata di due moduli di memoria *SDRAM*, modello *MT48LC4M16A2*, collegati in parallelo, formando in tal modo un bus di 32 bit, che quindi dota il sistema di 16 Mbytes di memoria RAM. Nell' appendice C si può vedere lo schema di connessione della memoria della piattaforma *Antares*.

Nella figura ***Figura A.4*** possiamo vedere in dettaglio i due moduli di memoria installati nella scheda *Antares*.

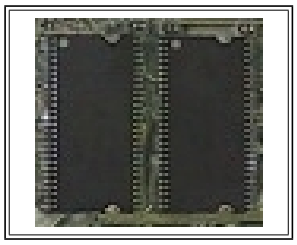


Figura A.4: *Dettaglio della memoria SDRAM*

A.2.2 Memoria Flash

Per il salvataggio di dati, in maniera non volatile, *Antares* utilizza memoria di tipo *Flash*. La scheda è stata progettata per essere dotata di una quantità di 2 o 4 Mbytes di memoria *Flash*. Questa decisione deriva dal fatto che, schede per applicazioni che non richiedano una grande quantità di spazio per immagazzinare i file binari che verranno eseguiti, possano essere fabbricate con solo uno dei due chip di memoria

Flash da 2 Mbytes, riducendo così il costo del sistema. Per rendere ciò possibile, si utilizzano due linee distinte di *chip-select*, per selezionare il relativo modulo di memoria *Flash*. Le linee che si utilizzano a questo scopo sono *CS0* e *CS1*.

Dei due moduli di memoria risulta obbligatorio installare quello connesso al *CS0* in quanto il processore *MCF5272* attiva detta linea nella fase di inizializzazione, ottenendo, dal dispositivo di memoria connesso a questa linea, il puntatore di stack e la direzione della prima istruzione che deve essere eseguita. Risulta perciò chiaro che il codice che viene eseguito nella fase di inizializzazione del sistema, deve essere immagazzinato nel dispositivo di memoria connesso con la linea *CS0*. D'ora in avanti ci riferiremo ai due moduli di memoria indicando semplicemente *Flash0* o *Flash1*, sottintendendo la linea di *chip-select* alla quale sono connessi.

Entrambi i moduli di memoria sono assolutamente identici. Il modello di memoria utilizzato è il *MBM29LV160BE* del produttore *Fujitsu* [3].

Questa famiglia di dispositivi è caratterizzata da tempi di accesso di *70ns* o *90ns*, a seconda del modello selezionato. Quelli installati in *Antares* hanno un tempo di accesso di emph90ns. Nel nostro sistema la memoria *Flash* è configurata per utilizzare i 16 bits del bus di dati. Per maggiori dettagli su come è stata realizzata la connessione tra i moduli di memoria *Flash* e il microcontrollore *MCF5272* si consultino gli schemi elettrici di *Antares* riportati in appendice C.

Nella ***Figura A.5*** possiamo vedere in dettaglio la zona della scheda *Antares* nella quale si trovano situati i dispositivi di memoria *Flash*.

Figura A.5: *Dettaglio della memoria Flash*

A.3 Interfacce

Con la denominazione di interfacce della scheda *Antares* indichiamo tutte le parti della scheda stessa, che le permettono la comunicazione con ciò a cui sarà connessa. *Antares* utilizza queste interfacce di

comunicazione, tanto con l'utente, quanto con i dispositivi che deve controllare.

In un sistema embedded nel quale le comunicazioni giocano un ruolo fondamentale, la disponibilità di molteplici interfacce di comunicazione è un fattore determinante. Segue una descrizione dettagliata di tutte le interfacce di cui questa scheda è dotata.

A.3.1 Interfaccia Ethernet

L'interfaccia *Ethernet* di *Antares* si compone di quattro parti: il controller, integrato nel *MCF5272*, che si incarica di realizzare l'accesso alla rete (*MAC*), un circuito integrato connesso al *MCF5272*, che offre le funzioni di livello fisico dell'interfaccia, alcuni traslatori di livello, che effettuano la conversione dei livelli di tensione, adattandoli alle specifiche e dal connettore *RJ-45* al quale si può connettere un convenzionale cavo di rete. Riassumendo, i blocchi dell'interfaccia *Ethernet* sono i seguenti:

1. *Controller di acceso MAC (MCF5272).*
2. *Transceptor DP83846A.*
3. *Traslatori di livello.*
4. *Connettore RJ-45.*

Nella figura ***Figura A.6*** vediamo l'ubicazione dei differenti componenti sulla piattaforma *Antares*.

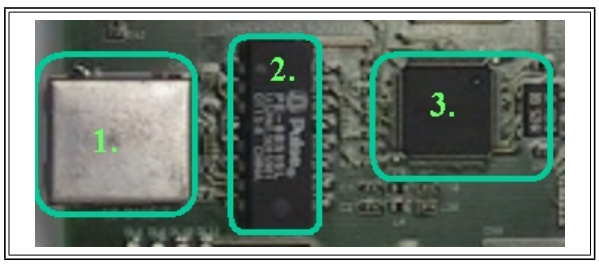


Figura A.6: *Dettaglio dell'interfaccia Ethernet*

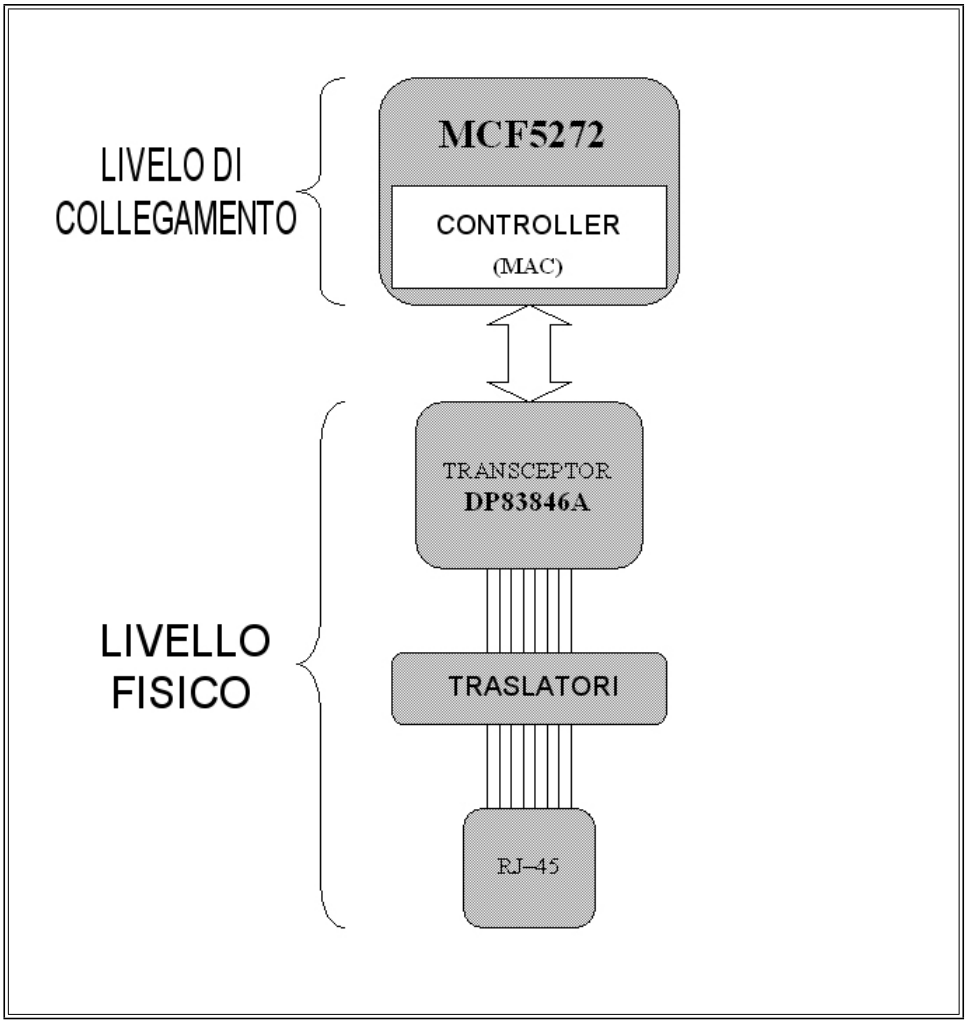


Figura A.7: *Diagramma a blocchi dell'interfaccia Ethernet*

La funzione di ciascun modulo in riferimento alla gerarchia dei protocolli *OSI* è descritta nella ***Figura A.7***. In questa figura possiamo vedere chiaramente che il controller integrato nel *MCF5272* realizza unicamente la funzione relativa al livello di collegamento, ovvero gestisce l'accesso livello fisico (*MAC*) e quindi necessita di un transceptor esterno che implementi le funzioni del livello fisico dell'interfaccia. Anche i traslatori di livello fanno parte dell'hardware del livello fisico, così come il connettore *RJ-45*.

A.3.2 Interfaccia USB

Il protocollo di comunicazione *USB* definisce due entità distinte nella comunicazione: controllore e dispositivo. Il controllore è responsabile della gestione della comunicazione, dando i permessi ai differenti dispositivi a lui connessi. Il controller integrato nel *MCF5272* si comporta da "dispositivo"; ciò significa che la scheda può essere connessa a periferiche *USB*, però non può essere usata come "controllore".

A differenza di quanto accade con il controller *Ethernet*, il controller *USB* del *MCF5272* è già dotato di un transceptor interno che permette la connessioe diretta ad una linea *USB*. Nonostante ciò, il microcontrollore è predisposto anche alla connessione di un transceptor esterno qualora il progettista ritenga necessario usare questo componente. Nel nostro caso abbiamo optato per l'utilizzo del transceptor interno, di modo da ridurre il numero di dispositivi della scheda e contribuendo quindi alla riduzione delle dimensioni della stessa. L'unico componente hardware che è stato necessario aggiungere è una sorgente di clock esterna, alla frequenza di *48 MHz*, frequenza utilizzata per l'invio e la ricezione dei dati.

Nella ***Figura A.8*** possiamo vedere in dettaglio l'ubicazione dei componenti dell'interfaccia *USB*

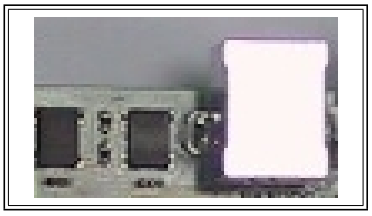


Figura A.8: *Detaglio dell'interfaccia USB*

A.3.3 Interfaccia seriale RS-232

Come già è stato visto anteriormente, il microcontrollore *MCF5272* possiede integrate due *UART* complete; esse permettono la comunicazione con altri dispositivi mediante una linea seriale. Questi dispositivi possono essere altri sistemi embedded, un terminale, una console, etc... É importante notare che le *UART* del microcontrollore lavorano con livelli di tensione *LCMOS* (*0 - +3.3V*), quindi, se desideriamo che risultino compatibili con i livelli di tensione *RS-232* (*-12V - +12V*), dovremo aggiun-

gere un elemento che si incarichi della trasformazione dei livelli di tensione. Questa è la funzione degli integrati *ICL3232*.

Gli ultimi componenti di questa interfaccia sono i connettori. Questi sono del tipo *DB-9* macho, uguali a quelli che si trovano su un normale personal computer. L'intenzione é chiaramente quella di poter connettere ad *Antares* un qualsiasi dispositivo, come per esempio un modem, nello stesso modo col quale potrebbe essere connesso ad un PC. Questo comporta che, se si vuole effettuare la connessione tra la scheda *Antares* e un personal computer, occorre usare un cavo "null-modem" di modo che i segnali *Rx*/*Tx* e *RTS*/*CTS* risultino correttamente connessi.

Nella ***Figura A.9*** possiamo leggere la piedinatura del connettore *DB9*.

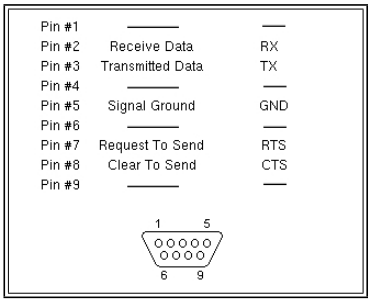


Figura A.9: *Segnali dell'interfaccia RS232*

Come già detto, *Antares* dispone di due interfacce *RS-232*, che denomineremo *Serie 0* o *Serie 1*, in funzione della porta *UART* del microcontroller alla quale sono connesse. L'ubicazione di dette interfacce nella piattaforma *Antares* può essere vista nella ***Figura A.10***.

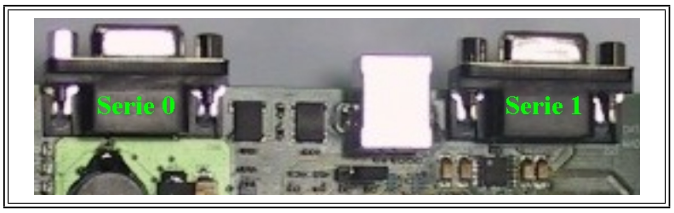


Figura A.10: *Dettaglio delle interfacce RS-232*

A.3.4 Entrate/uscite digitali

La piattaforma *Antares* è dotata di due porte digitali da 8 bit cada una. Di queste due, solo una è bidirezionale e sarà d'ora in avanti chiamata *Porta 1*. L'altra porta

è esclusivamente di uscita e sarà denominata *Porta 0*. Nella ***Figura A.11*** possiamo notare la collocazione di queste due porte nella scheda, così come la posizione del bit più significativo di ciascuna di queste.

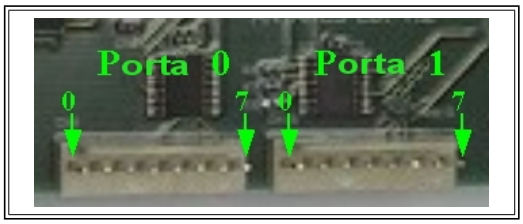


Figura A.11: *Dettaglio delle porte di entrata/uscita*

La *Porta 0* utilizza una delle linee di *chip-select*, la *CS2* per abilitare un buffer di uscita che immagazzina il valore presente nel bus dati in un dato momento. Questo sarà il valore che verrà letto dalla porta quando il sistema operativo effettua l'operazione di lettura.

La *porta 1* utilizza, invece, una delle porte di entrata/uscita del *MCF5272*. Per essere utilizzata questa porta andrà configurata, via software, in maniera che si adatti alle nostre necessità, in quanto permette la configurazione in forma indipendente, di ciascuno dei pin del connettore (entrata o uscita).

Entrambe le porte lavorano con livelli di tensione *LCMOS*, ovvero *3.3V*, e quindi, all'atto di connettere la nostra scheda con altri dispositivi, dovremo badare alla compatibilità tra i livelli logici. Nella ***Tabella A.1*** sono illustrati i livelli di tensione delle famiglie logiche più comuni.

Parametro	**LCMOS (3.3V)**	**HCMOS (5V)**	**TTL (5V)**	**TTL (3.3V)**
$V_{IH_{min}}$	2 V	3.5V	2V	2V
$V_{IL_{max}}$	0.8 V	0.8 V	0.8V	0.8V
$V_{OH_{min}}$	2.4 V	4.44 V	2.4V	2.4V
$V_{OL_{max}}$	0.5 V	0.5 V	0.4V	0.4V

Tabella A.1: *Livelli di tensione delle famiglie logiche più comuni*

Per conoscere maggiori dettagli tecnici in merito alle porte di *ingresso/uscita* qui illustrate, si rimanda agli schemi di *Antares* nell'appendice C.

A.3.5 Interfaccia *BDM/JTAG*

Antares dispone di un'interfaccia *BDM/JTAG*. Con il jumper posto vicino all connettore, si può selezionare il tipo di protocollo usato da questa interfaccia. Ai nostri fini useremo il *BDM* che permette, tramite un adattatore esterno, di connettere la scheda alla porta parallela di un personal computer. Tramite questo collegamento, e appositi driver e programmi per gestire il protocollo *BDM*, potremo:

- scrivere dei dati nella memoria *Flash* della scheda. Questo è l'unico modo per scrivere nella memoria *Flash* quando sulla scheda non è ancora presente un *firmware* o quando il sistema operativo, per qualche motivo, non riesca ad avviarsi.
- effettuare il debug degli applicativi caricati in memoria senza dover ricorrere a costosi simulatori o emulatori.

Nella ***Figura A.12*** si può vedere la posizione del connettore e quella del relativo jumper di configurazione.

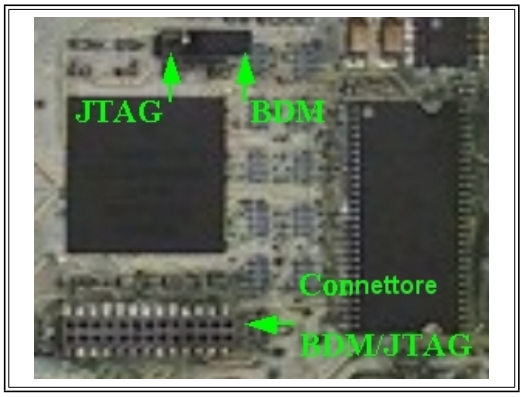


Figura A.12: *Dettaglio dell'interfaccia BDM/JTAG*

A.3.6 Bus di espansione

Il bus di espansione di *Antares* è formato da due connettori, *CN2* e *CN3*, posizionati parallelamente su uno dei lati della scheda. In questi connettori sono state portate le linee dati più importanti del *MCF5272*, per permettere l'espansione della scheda tramite schede aggiuntive. Tra questi segnali troviamo il bus di dati, quello di indirizzamento, le linee di interrupt, le linee del banco di *SDRAM*,etc...

Nella ***Figura A.13*** sono mappati i segnali presenti sui due connettori della scheda.

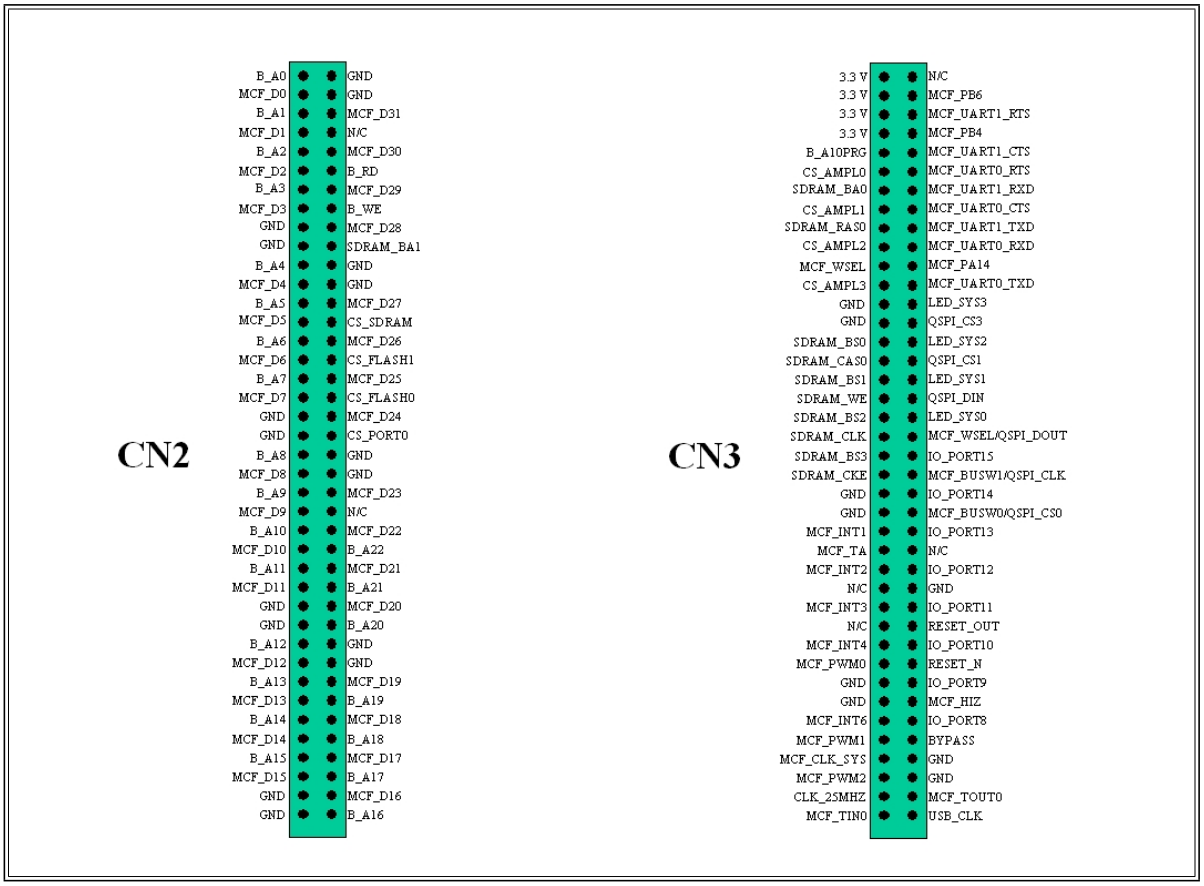


Figura A.13: *Segnali del bus di espansione*

Come si può vedere dettagliatamente nella figura, al connettore *CN2* sono state portate le linee del bus di dati, del bus di indirizzamento, le linee del banco di *SDRAM*, le linee di *chip-select* della memoria *Flash* e della *Porta 0* e molteplici connessioni di massa.

Nel connettore *CN3* vi sono i rimanenti segnali del bus, vale a dire le linee di alimentazione da *3.3V* e massa, le linee di interrupt esterne, i *chip-select* non usati, i segnali del bus *SPI*, le linee delle due *UARTs*, i temporizzatori ,le linee dei modulatori per ampiezza di impulsi (PWM)...

La ***Figura A.14*** mostra l'ubicazione dei due connettori di espansione sulla scheda *Antares*.

Figura A.14: *Dettaglio del bus di espansione*

A.4 Alimentazione

La scheda *Antares* può essere alimentata con una qualsiasi sorgente di tensione continua che fornisca un valore di tensione compreso tra *8V* e *14V*. Questa tensione verrà trasformata dallo stadio di alimentazione, in una tensione continua stabilizzata di *3.3V*, che è il valore di tensione alla quale lavora il microcontrollore *MCF5272*. Per ottenere i sudetti *3.3V* è stato progettato uno stabilizzatore di tensione commutato[4] molto efficiente e molto più adatto di un normale stabilizzatore lineare, soprattutto se si tiene in conto che la scheda *Antares* è mirata alla realizzazione di sistemi embedded, che devono essere per definizione molto parsimoniosi in termini di consumo di corrente.

[4]Gli stabilizzatori di tensione commutati possono raggiungere efficienze dell'*85%* e possono lavorare con un rango di tensioni d'ingresso molto più amplio

Nella ***Figura A.15*** è illustrato lo stadio di alimentazione della scheda *Antares*.

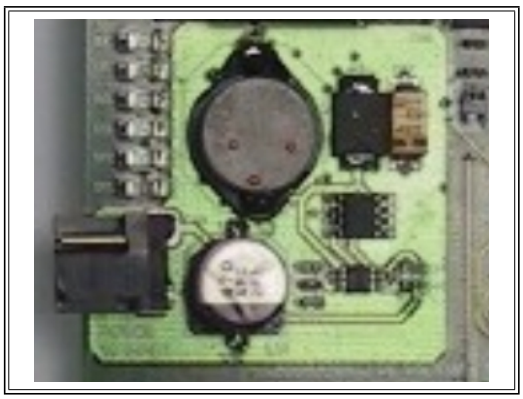


Figura A.15: *Dettaglio dello stadio di alimentazione*

Appendice B

Bibliografia

Bibliografía

[1] AMD. *AMD Semiconductor Data Sheet*.

[2] Bernardo Cascales Salinas, Pascual Lucas Saorín, Jose Manuel Mira Ros, Antonio Pallarés Ruiz, and Salvador Sánchez-Pedreño Guillén. *LaTeX una imprenta en sus manos*. Aula Documental de Investigación, 2000.

[3] Fujitsu. *Fujitsu Semiconductor Data Sheet*.

[4] Josè David Romeral Martìn Juan Manuel Montero Martìnez, Javier Macìas Guarasa. *Enunciado de la pràctica estandar del Laboratorio de Sistemas Electrònicos Digitales (LSED)*. ETSIT, 2001.

[5] Motorola. *dBUG Reference Manual*, 1999.

[6] Motorola. *Coldfire Family Programmer's Reference Manual*, 2001.

[7] Motorola. *MCF5272 Coldfire Integrated Microprocessor User's Manual*, 2001.

[8] Alessandro Rubini. *Linux Device Drivers*. O'Reilly & Associates, primera edition, 1998.

[9] Juan Jose San Martín. uclinux empotrado en arquitecturas digitales basadas en microcontroladores y lógica programable. Master's thesis, ETSIT, 2001.

Riferimenti Internet

[10] URL: www.uclinux.org - Pagina web della distribuzione uClinux.

[11] URL: www.cygwin.com - Pagina web di Cygwin.

[12] URL: www.ucdot.org - Embedded Linux Developer Forum.

[13] URL: e-www.motorola.com - Pagina web di Motorola Semiconductors.

[14] URL: java.sun.com/j2me/ - Java 2 Platform, Micro Edition (J2ME).

[15] URL: http://www.fiddes.net/coldfire/ - The Linux Documemtation Project.

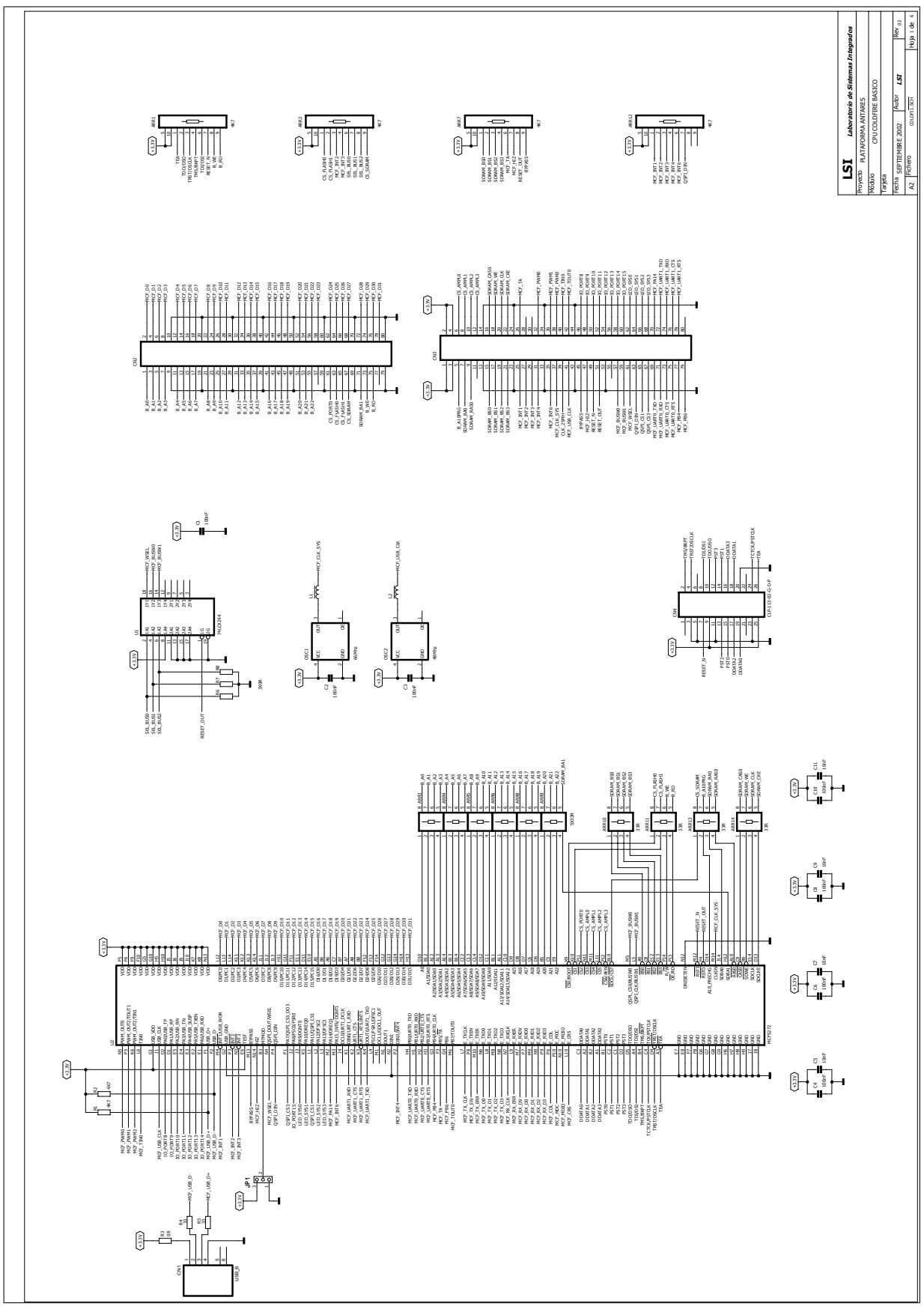
[16] URL: http://www.fiddes.net/coldfire/ - Pagina web del compilatore per microcontrollori ColdFire.

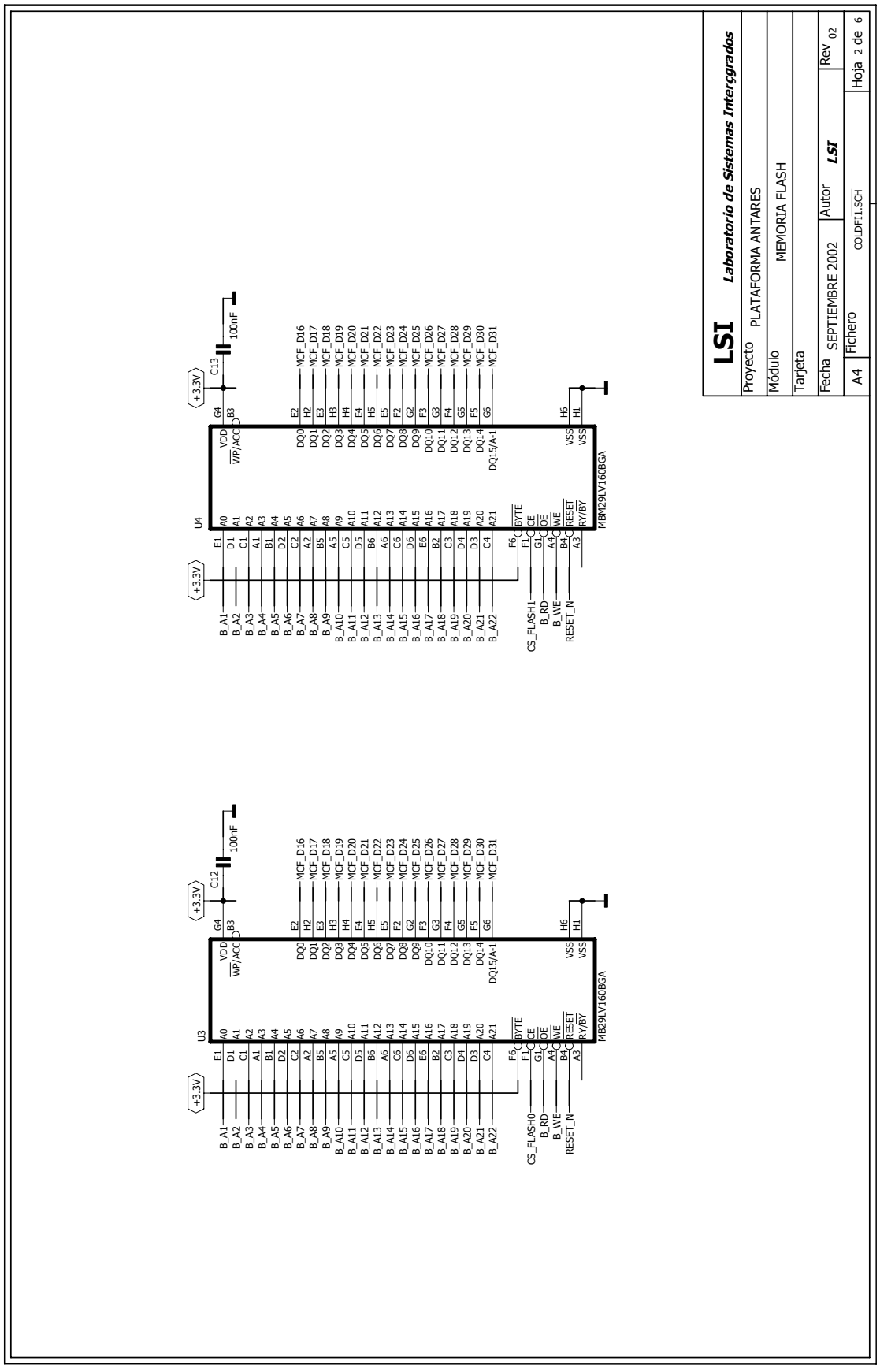
[17] URL: gcc.gnu.org/ - Pagina web del GCC (GNU Compiler Collection).

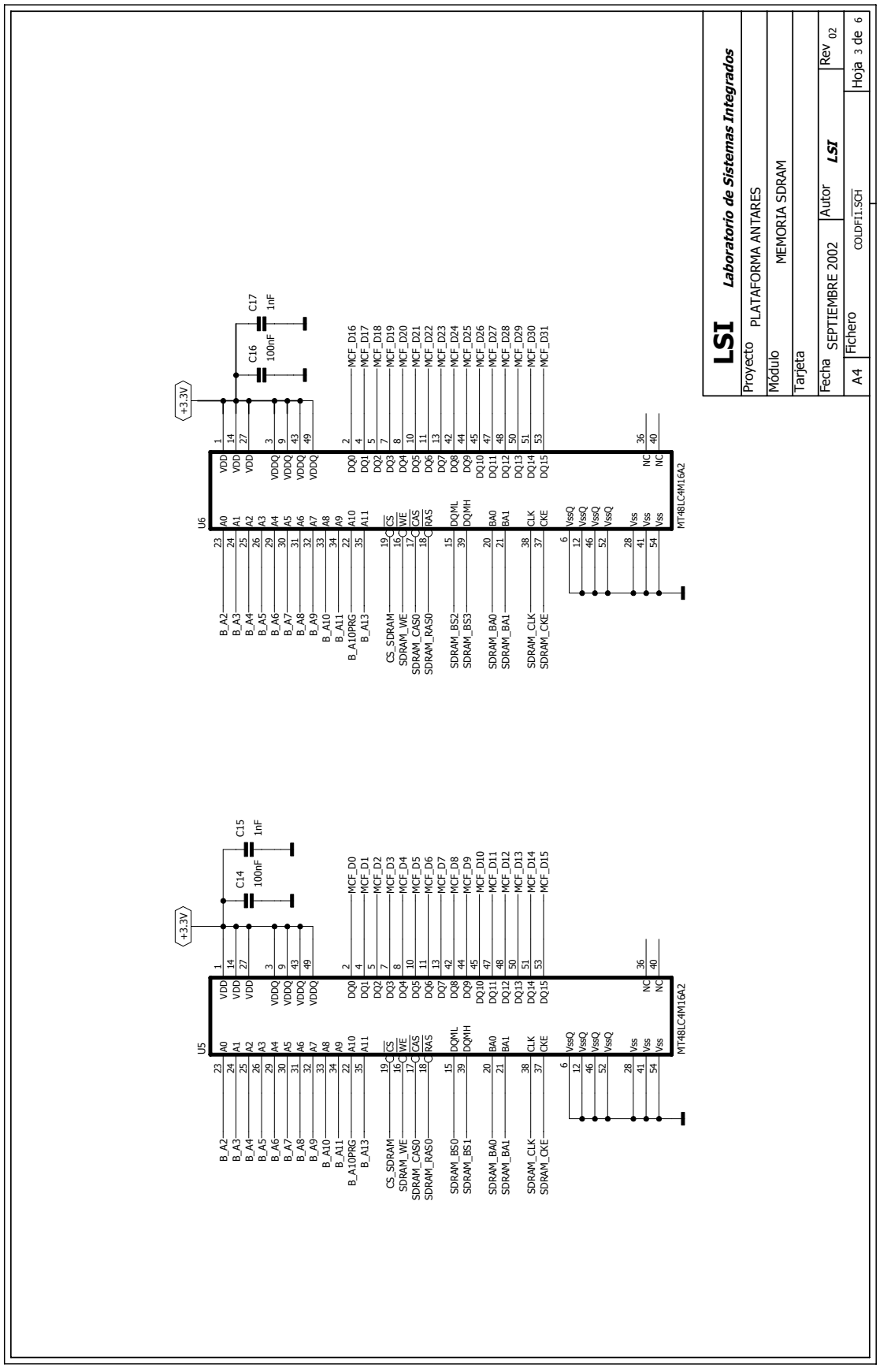
[18] URL: http://www.gnu.org/copyleft/gpl.html - GNU General Public License.

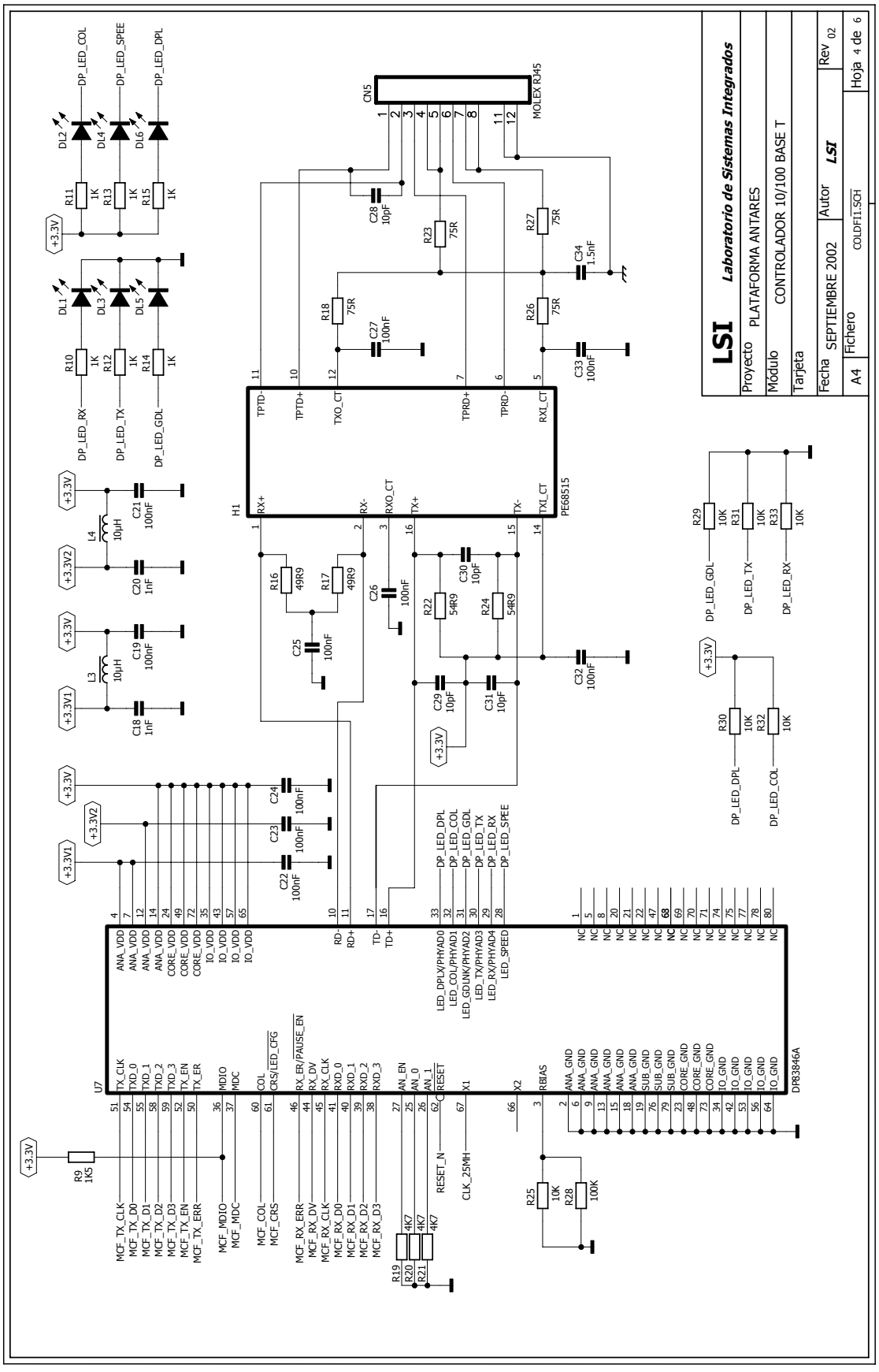
Appendice C

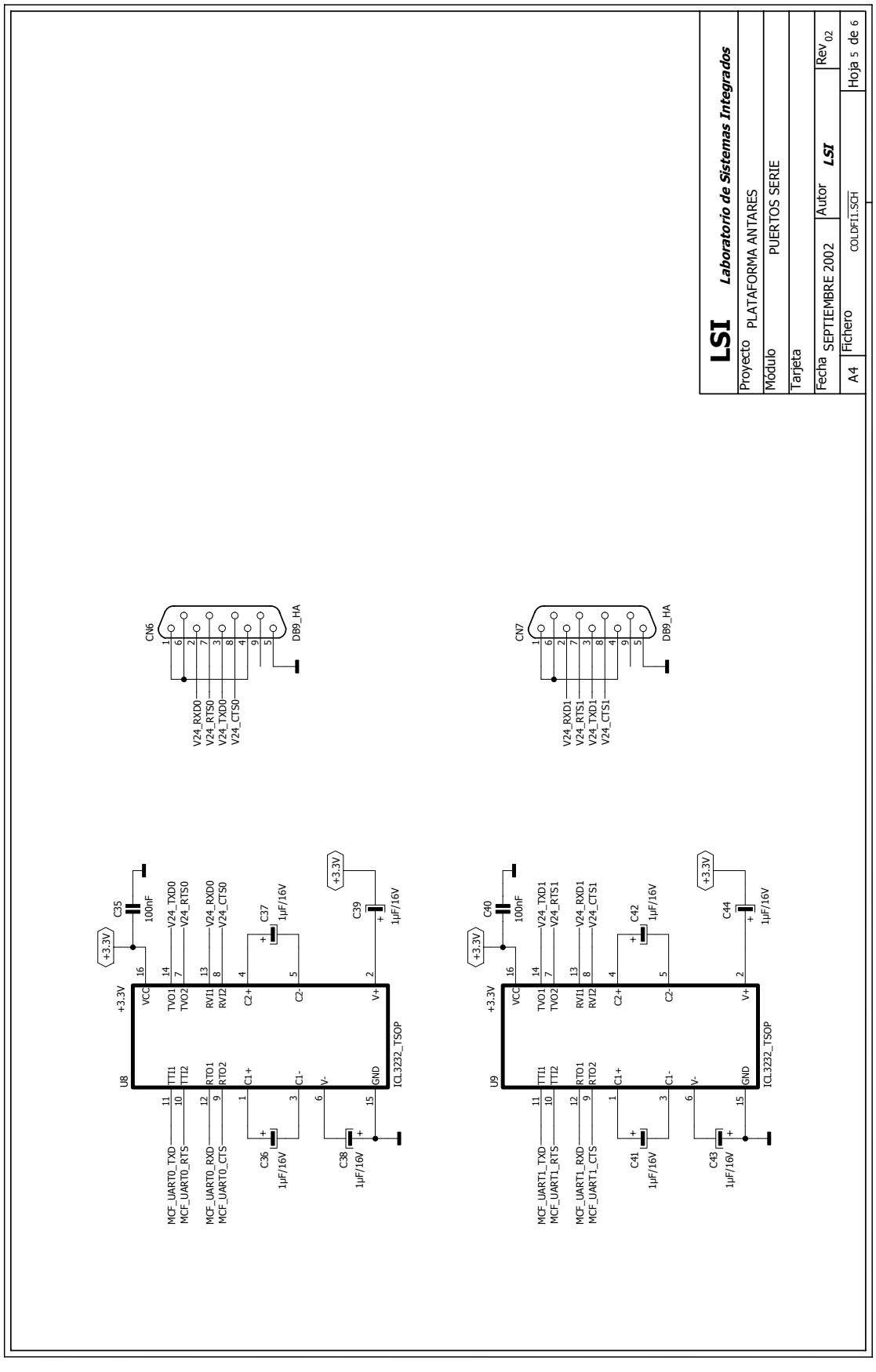
Schemi elettrici di *Antares*

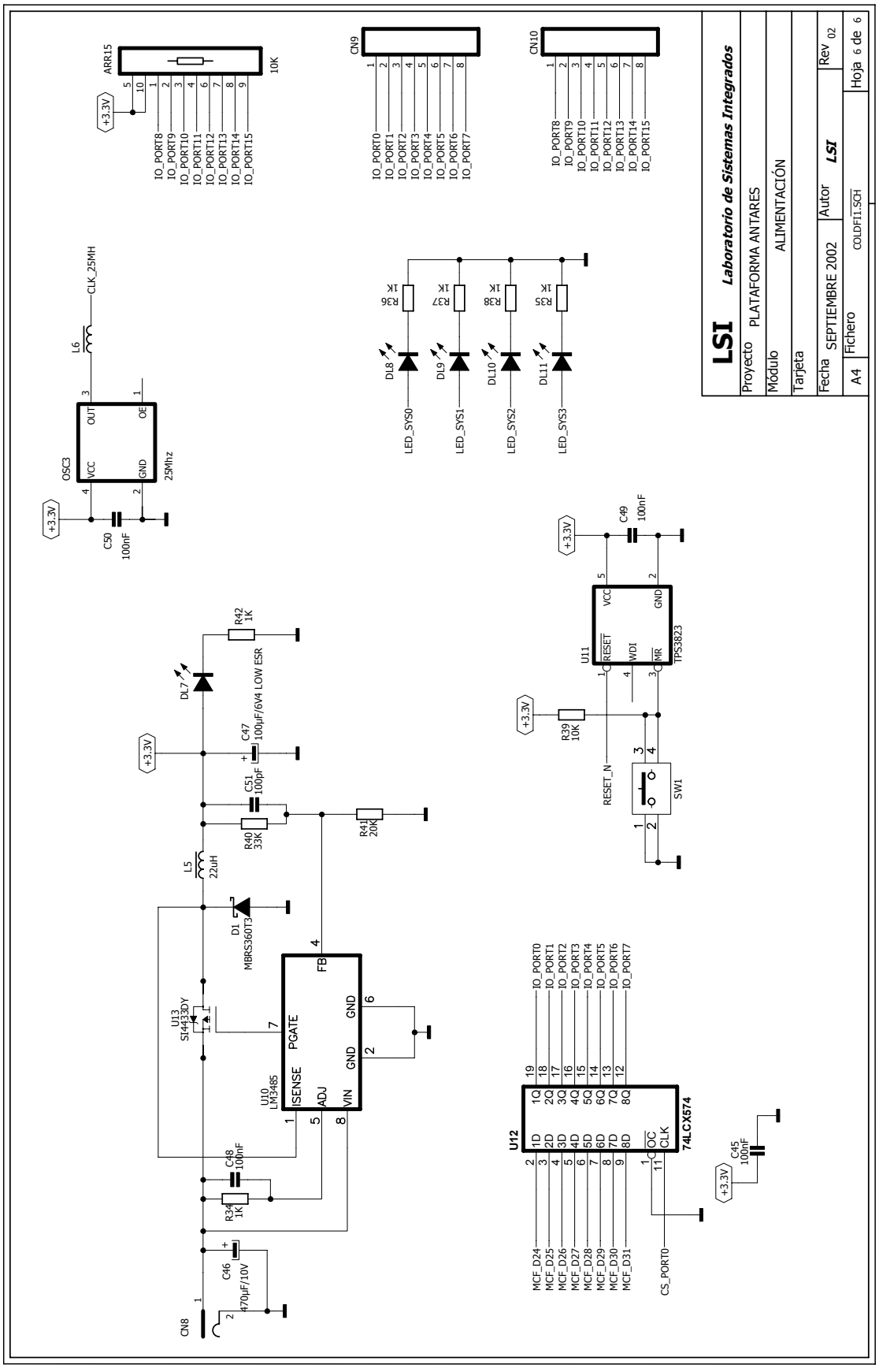

LSI
Laboratorio de Sistemas Integrados
Proyecto PLATAFORMA ANTARES
Modulo CPU COLDFIRE BASICO
Tarjeta
Fecha SEPTIEMBRE 2002
Autor LSI
Rev 0.1
A2 Fichero cpu.prt.SCH
Hoja 1 de 6


LSI Laboratorio de Sistemas Interçgrados
Proyecto PLATAFORMA ANTARES
Módulo MEMORIA FLASH
Tarjeta
Fecha SEPTIEMBRE 2002
Autor LSI
Rev 02
A4
Fichero COLDFI1.SCH
Hoja 2 de 6
U3
U4
MBM29LV160BGA
+3.3V
C12
C13
100nF
B_A1
B_A2
B_A3
B_A4
B_A5
B_A6
B_A7
B_A8
B_A9
B_A10
B_A11
B_A12
B_A13
B_A14
B_A15
B_A16
B_A17
B_A18
B_A19
B_A20
B_A21
B_A22
CS_FLASH0
CS_FLASH1
B_RD
B_WE
RESET_N
MCF_D16
MCF_D17
MCF_D18
MCF_D19
MCF_D20
MCF_D21
MCF_D22
MCF_D23
MCF_D24
MCF_D25
MCF_D26
MCF_D27
MCF_D28
MCF_D29
MCF_D30
MCF_D31


LSI Laboratorio de Sistemas Integrados
Proyecto PLATAFORMA ANTARES
Módulo MEMORIA SDRAM
Tarjeta
Fecha SEPTIEMBRE 2002
Autor LSI
Rev 02
A4
Fichero COLDFI1.SCH
Hoja 3 de 6
+3.3V
C17 1nF
C16 100nF
C15 1nF
C14 100nF
U6
U5
MT48LC4M16A2
B_A2
B_A3
B_A4
B_A5
B_A6
B_A7
B_A8
B_A9
B_A10
B_A11
B_A10PRG
B_A13
CS_SDRAM
SDRAM_WE
SDRAM_CAS0
SDRAM_RAS0
SDRAM_BS2
SDRAM_BS3
SDRAM_BA0
SDRAM_BA1
SDRAM_CLK
SDRAM_CKE
MCF_D16
MCF_D17
MCF_D18
MCF_D19
MCF_D20
MCF_D21
MCF_D22
MCF_D23
MCF_D24
MCF_D25
MCF_D26
MCF_D27
MCF_D28
MCF_D29
MCF_D30
MCF_D31
MCF_D0
MCF_D1
MCF_D2
MCF_D3
MCF_D4
MCF_D5
MCF_D6
MCF_D7
MCF_D8
MCF_D9
MCF_D10
MCF_D11
MCF_D12
MCF_D13
MCF_D14
MCF_D15


LSI Laboratorio de Sistemas Integrados
Proyecto PLATAFORMA ANTARES
Módulo CONTROLADOR 10/100 BASE T
Tarjeta
Fecha SEPTIEMBRE 2002
Autor LSI
Rev 02
A4
Fichero COLDFI1.SCH
Hoja 4 de 6
U7
DP83846A
H1
PE68515
CN5
MOLEX RJ45


LSI Laboratorio de Sistemas Integrados
Proyecto PLATAFORMA ANTARES
Módulo PUERTOS SERIE
Tarjeta
Fecha SEPTIEMBRE 2002
Autor LSI
Rev 02
A4
Fichero COLDFI1.SCH
Hoja 5 de 6
CN6
DB9_HA
V24_RXD0
V24_RTS0
V24_TXD0
V24_CTS0
CN7
DB9_HA
V24_RXD1
V24_RTS1
V24_TXD1
V24_CTS1
U8
ICL3232_TSOP
U9
ICL3232_TSOP
+3.3V
C25
100nF
C36
1µF/16V
C37
1µF/16V
C38
1µF/16V
C39
1µF/16V
C40
100nF
C41
1µF/16V
C42
1µF/16V
C43
1µF/16V
C44
1µF/16V
MCF_UART0_TXD
MCF_UART0_RTS
MCF_UART0_RXD
MCF_UART0_CTS
MCF_UART1_TXD
MCF_UART1_RTS
MCF_UART1_RXD
MCF_UART1_CTS


LSI Laboratorio de Sistemas Integrados
Proyecto PLATAFORMA ANTARES
Módulo ALIMENTACIÓN
Tarjeta
Fecha SEPTIEMBRE 2002
Autor LSI
Rev 02
A4
Fichero COLDFI1.SCH
Hoja 6 de 6
ARR15
10K
CN9
CN10
IO_PORT0
IO_PORT1
IO_PORT2
IO_PORT3
IO_PORT4
IO_PORT5
IO_PORT6
IO_PORT7
IO_PORT8
IO_PORT9
IO_PORT10
IO_PORT11
IO_PORT12
IO_PORT13
IO_PORT14
IO_PORT15
+3.3V
OSC3
CLK_25MH
25Mhz
C50 100nF
L6
DL8
DL9
DL10
DL11
R35 1K
R36 1K
R37 1K
R38 1K
LED_SYS0
LED_SYS1
LED_SYS2
LED_SYS3
U11
TPS3823
RESET
WDI
MR
VCC
GND
C49 100nF
R39 10K
SW1
RESET_N
U13
SI4433DY
U10
LM3485
PGATE
ISENSE
ADJ
VIN
FB
GND
L5 22uH
D1 MBRS360T3
DL7
R42 1K
C47 100µF/6V4 LOW ESR
C51 100pF
R40 33K
R41 20K
C48 100nF
R34 1K
C46 470µF/10V
CN8
U12
74LCX574
1D
2D
3D
4D
5D
6D
7D
8D
1Q
2Q
3Q
4Q
5Q
6Q
7Q
8Q
OC
CLK
MCF_D24
MCF_D25
MCF_D26
MCF_D27
MCF_D28
MCF_D29
MCF_D30
MCF_D31
CS_PORT0
C45 100nF